高等学校计算机应用规划教材

ASP.NET 4.0 网站开发实例教程

◆ ASP.NET内置对象
◆ 服务器控件
◆ 站点导航
◆ 页面布局与主题应用
◆ 数据绑定与数据显示
◆ AJAX局部刷新
◆ Web服务
◆ LINQ查询
◆ Web站点的安全性
◆ jQuery实现动画效果



耿超　编著

清华大学出版社

高等学校计算机应用规划教材

ASP.NET 4.0 网站开发实例教程

耿超　编著

清华大学出版社
北　京

内 容 简 介

本书全面翔实地介绍了使用 ASP.NET 4.0 开发动态网站的基本知识和实用技巧。全书共分 12 章，主要内容包括：ASP.NET 的发展历程，VWD 2010 集成开发环境，ASP.NET 的内置对象和配置管理，服务器控件的使用，使用 CSS、主题和母版页进行页面设计与布局，使用 ADO.NET 访问数据库和操纵 XML，数据绑定与数据控件的使用，使用 LINQ 查询，Web 站点的安全性，ASP.NET AJAX 技术，Web 服务的创建与调用，使用 jQuery 美化网页以及站点的发布与部署，最后一章是综合运用所学知识创建了一个简易微博系统。本书重点介绍了常用控件的使用以及动态网站开发的实用技巧，并安排了多个综合性的个性网站开发实例。此外，每章还配有习题，有助于读者对所学知识的理解与掌握，提高和拓宽读者的实际技能。

本书结构清晰、内容翔实，既可以作为高等院校本科学生的专业教材，也可以作为从事网站开发与设计工作的专业技术人员的参考书。

本书的电子教案、实例源文件和习题答案可以到 http://www.tupwk.com.cn/downpage/index.asp 网站下载。

图书在版编目(CIP)数据

ASP.NET 4.0 网站开发实例教程/耿超 编著. —北京：清华大学出版社，2013.1（2018.1 重印）
(高等学校计算机应用规划教材)
ISBN 978-7-302-30496-8

I. ①A… II. ①耿… III. ①网页制作工具—程序设计—高等学校—教材 IV. ①TP393.092

中国版本图书馆 CIP 数据核字(2012)第 257015 号

责任编辑：胡辰浩 袁建华
装帧设计：康 博
责任校对：邱晓玉
责任印制：沈 露

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦 A 座 邮 编：100084
社 总 机：010-62770175 邮 购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tupwk.com.cn，010-62794504
印 装 者：北京九州迅驰传媒文化有限公司
经 销：全国新华书店
开 本：185mm×260mm **印 张：**24.5 **字 数：**581 千字
版 次：2013 年 1 月第 1 版 **印 次：**2018 年 1 月第 2 次印刷
印 数：5001～5500
定 价：36.00 元

产品编号：043716-01

前　　言

ASP.NET 是 Microsoft 公司推出的基于.NET Framework 的 Web 应用开发平台，是 Web 应用开发的主流技术之一，它带给人们的是全新的技术，和由此产生的开发效率的提高，网站性能的提升。使用 ASP.NET 进行 Web 应用开发，程序结构更加清晰，开发流程更加简单，从而可以提高开发效率，缩短开发周期。ASP.NET 4.0 是在 ASP.NET 3.5 的基础之上构建的，保留了其中很多令人喜爱的功能，并增加了一些其他领域的新功能和工具。

进行 ASP.NET 开发可以使用 Visual Basic.NET 或者 C#，这两种语言都是.NET 环境下的程序设计语言，但并不是必须使用.NET 集成开发环境才能进行 ASP.NET Web 程序设计。从理论上讲，用记事本或其他文本编辑器就可以编写 ASP.NET Web 应用程序，但大多数开发人员还是希望安装 Microsoft Visual Web Developer 2010(VWD)。VWD 是专门为构建 ASP.NET Web 站点而开发的，其中包含了大量有助于快速创建复杂 ASP.NET Web 应用程序的工具。为了使广大学生和网站开发技术人员尽快掌握 ASP.NET 4.0 以及 VWD 2010 的使用技巧和新增功能，本人在多年教学经验与科研成果的基础上编写了此书。本书全面翔实地介绍了 ASP.NET 动态网站开发的基本流程和使用方法，可以使读者快速、全面地掌握使用 VWD 2010 快速开发 ASP.NET Web 应用程序，并达到融会贯通，灵活应用之目的。

概括起来，本书具有以下主要特点。

(1) 结构清晰、内容翔实。在每一章的开始概要说明了本章将介绍的内容，使学习者做到心中有数；介绍每一个内置对象或服务器控件时，首先介绍该对象或控件的常用属性、方法和事件，然后介绍具体的应用场景和注意事项，且在介绍过程中都配有插图和实例说明；在每章的最后还有对应的本章小结，总结本章介绍的内容，前后呼应，系统性强。

(2) 按照动态网站的开发流程，从网站基础知识讲起，循序渐进地介绍了 ASP.NET 动态网站开发涉及到的各种知识和技巧，并在各章配有精心选择的应用实例，这些实例既有较强的代表性和实用性，又能够综合应用对应章节介绍的知识，使学习者能够全方位地理解所学内容，融会贯通章节之间的联系，达到举一反三之目的。

(3) 每一章最后都配有习题。这些习题紧扣该章介绍的内容。通过回答和练习这些习题，有助于读者更好地掌握本章所学知识，提高和拓宽读者的实际技能。

全书共分 12 章，第 1 章介绍了网站建设的基础知识、ASP.NET 的发展过程、VWD 2010 开发环境，以及 ASP.NET 的工作原理；第 2 章介绍了 ASP.NET 的页面框架和页面类，ASP.NET 的内置对象以及 ASP.NET 的配置管理；第 3 章介绍了 ASP.NET 服务器控件的基本用法以及不同类别控件的功能；第 4 章介绍了 CSS 样式、主题和母版页，这些技术对于页面布局、创建具有一致外观的网站非常有用，也有利于使站点看起来更专业和有吸引力；第 5 章介绍了数据库的基本知识和 SQL 语言、使用 ADO.NET 访问数据库、数据绑定技术

和数据控件的使用以及使用 ADO.NET 读写 XML 文件；第 6 章介绍了 LINQ 语言及其语法；第 7 章介绍了 Web 站点中的安全性和 ASP.NET 提供的登录控件的用法；第 8 章介绍了 ASP.NET AJAX 的基本知识以及 ASP.NET AJAX 服务器控件的使用方法；第 9 章介绍了 Web 服务的基本概念以及如何创建和调用 Web 服务，包括在 AJAX 站点中使用 Web 服务；第 10 章介绍了 jQuery 的基本语法和具体应用，jQuery 能够改变我们编写 JavaScript 脚本的方式，降低学习和使用 Web 前端开发的复杂度，提高网页开发效率；第 11 章介绍了 Web 应用程序的发布与部署，包括复制 Web 站点、在 IIS 下运行站点和将数据库移动到远程服务器；第 12 章介绍综合运用全书所学知识，开发一个简易的迷你微博系统。

本书是多人智慧的集成，除封面署名的作者外，参与本书编写的人员还有黄果、李琼琼、周倩芸、韩高洁和郭纳等。在本书的编写过程中，参考了一些有关文献，在此向这些文献的作者深表感谢。由于作者水平有限，本书不足之处在所难免，欢迎广大读者批评指正。我们的信箱：huchenhao@263.net，电话：010-62796045。

作　者

2012 年 9 月

目　　录

第1章　ASP.NET 4.0入门

ASP.NET 4.0 是微软公司为了迎接网络时代的到来而提出的一个 Web 开发模型，它是建立在公共语言运行库上的编程框架，支持 C＃ 和 VB.NET 语言。本章将主要介绍网站建设的基础知识、ASP.NET 的基本原理及其发展史、VWD 2010 开发环境，通过本章的学习，读者应该掌握如何安装、使用 ASP.NET 集成开发环境——VWD 2010，并能够创建简单的 Web 站点。

本章学习目标：

- 网站建设基础知识
- ASP.NET 的发展史
- VWD 2010 开发环境
- 新建 Web 站点
- ASP.NET 应用程序的工作原理

1.1　网站建设概述

互联网最早出现于 20 世纪 60 年代末，早期的互联网用户大多限于教育和国防机构。随着越来越多的用户在全球范围内实现信息共享，互联网也逐渐兴盛起来。互联网的快速发展给人们的工作、学习和生活带来了巨大变化，极大地提高了工作效率。在互联网的发展过程中，Web 网站的开发技术也得到了不断的发展，而最为关键的技术之一就是网站建设技术。本节将介绍网站制作过程中的静态网站和动态网站等一些基本概念。

1.1.1　HTML 语言

HTML 的英文全称是 Hyper Text Markup Language，直译为超文本标记语言，它由 W3C 组织商讨制定。HTML 不是一个程序语言，而是一种描述文档结构的标记语言。

HTML 文档是含有标记、文本和影响文本的附加数据的简单文本文件。HTML 与操作系统平台的选择无关，只要有 Web 浏览器就可以运行 HTML 文件，显示网页内容。

1. HTML 元素和标记

HTML 用尖括号间的文本指示内容在浏览器中如何显示。这种带有尖括号的文本称为标记(tag)；含有文本或其他内容的一对标记称为元素。例如：

```
<h2>你好</h2>
<p>欢迎学习 ASP.NET 4 </p>
```

该示例的第一行含有一个带起始标记<h2>和结束标记</h2>的<h2>元素。此元素用来表示二级标题。注意，元素的结束标记和起始标记相似，只是前面多了个斜杠(/)。起始标记和结束标记之间的所有文本都被看做是标题部分。在二级标题元素下面，是一个<p>元素，它用来表示段落。<p>标记对中的所有文本都被看做是段落部分。

HTML 中有许多可用的标记；在这里就不一一介绍了。表 1-1 列出了一些最重要的标记。

表 1-1　HTML 标记

标　记	说　明	示　例
<html>	用来表示整个页面的开始和结束	<html> ...All other content goes here </html>
<head>	用来表示包含页面数据的页面的特殊部分，包括其标题以及对外部资源的引用	<head> ... Content goes here </head>
<title>	用来定义页面的标题。这个标题将会显示在浏览器的标题栏中	<title> 首页 </title>
<body>	用来表示页面体的开始和结束	<body> Page body goes here </body>
<a>	用来将一个 Web 页面链接到另一个页面	 百度一下
<img>	用来向页面中嵌入图像	
 <i> <u>	用来将文本格式化为粗体、斜体或下划线字体	这是粗体 <i>这里是斜体</i>
<form> <input> <select>	用于描述允许用户向服务器提交信息的输入格式	<input type="text" value=" 请输入 " />
<table> <tr> <td>	这些标记用来创建含有表格的布局。<table>标记定义了整个表，而<tr>和<td>标记分别用来定义行和单元格	<table> <tr> <td>第一列</td><td>第二列</td> </tr> </table>

(续表)

标　记	说　明	示　例
	这 3 个标记用来创建带有编号或	
 	项目符号的列表。 和 标记定义了列表的外观，而 标记用来表示列表中的项	- 有编号的 1 - 有编号的 2
	这个标记用来包装和影响文档的其他部分。它作为内联文本出现，所以不会向页面添加换行符	普通文本 红色文本
<div>	与<span>标记一样，<div>标记用来作为其他元素的容器。默认情况下<div>元素后面出现显式的换行符	 This is some text on 1 line

2. HTML 属性

除了有 HTML 元素之外，还有 HTML 属性。这些属性包含了一些改变特定元素行为方式的额外信息。例如，使用<img>标记显示一个图像，src 属性用于定义图像的源代码。通常，不需要记住所有这些元素和属性。在大多数情况下，VWD 会自动地生成它们，当需要手工输入时，也会给出智能提示，帮助找到正确的标记或属性。

说明：

除了标记，在 HTML 中还常常引用脚本语言(Scripting Language)，如 JavaScript 和 VBScript。使用脚本语言，可以制作出网页特效和一些简单的动态效果。

3. HTML 和 XHTML 的区别

除了 HTML 之外，还有 XHTML。虽然两者的名称看起来非常相似，但是它们之间有一些有趣的区别。XHTML(eXtensible Hypertext Markup Language)称为可扩展超文本标记语言，是为了使 HTML 向 XML(eXtensible Markup Language)过渡而定义的标记语言，它以 HTML 为基础，采用 XML 严谨的语法结构。XML 是一种通用的、用来描述数据的、基于文本与标记的语言，也是作为其他许多语言(包括 XHTML)的基础语言。

XHTML 很大程度上是用 XML 规则重写的 HTML。在 XHTML 中，如果用<p>开始了一个段落，就必须在页面后面的某个地方用</p>闭合该段落。对于没有结束标记的标记也是如此，比如<img>或
(用来输入一个换行符)。在 XHTML 中，这些标记被写为自结束标记，其中结束标记中的斜杠直接嵌在标记自身中，例如：

```
<img src="Logo.gif" />或<br />
```

XML 是区分大小写的，XHTML 通过强制所有标记采用小写来应用该规则。虽然标记

和特性必须都是小写，但是实际值不必是这样。例如，前面显示 logo 图像的示例是完全有效的 XHTML，这里的图像名称中使用了大写的 L。

大部分的浏览器都可以正确解析 XHTML，即使老版本的浏览器，也将 XHTML 作为 HTML 的一个子集。

注意：

除了 HTML 之外，ASP.NET Web 页面也可能包含其他标记。大多数页面上都有一个或多个 ASP.NET 服务器控件，给出了一些附加的功能。本书第 3 章将深入介绍 ASP.NET 服务器控件。

1.1.2　静态网站

静态网站是指全部由 HTML 代码格式页面组成的网站，所有的内容包含在网页文件中，文件扩展名为：.htm、.html、.shtml、.xml 等。网页上也可以出现各种视觉动态效果，如 GIF 动画、FLASH 动画和滚动字幕等。

早期的网站一般都是采用静态网页技术制作的静态网站。这里所讨论的“静”是指网页内容在用户发出请求之前就已经生成了(也就是说，用户每次总能看到相同的页面)，Web 服务器只负责保存和传递 HTML 文件，而不进行额外处理，用户只能阅读网站所提供的信息，这种页面的请求模式如图 1-1 所示。

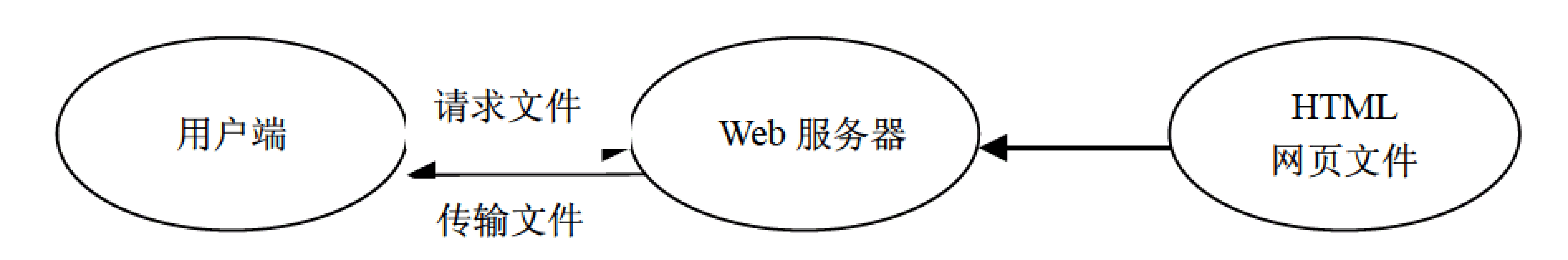


图 1-1　静态网站模型

可将静态网页的特点简要归纳如下。

(1) 静态网页每个网页都有一个固定的 URL，且网页 URL 以.htm、.html、.shtml 等常见形式为后缀。

(2) 网页内容一经发布到网站服务器上，无论是否有用户访问，每个静态网页的内容都是保存在网站服务器上的，每个网页都是一个独立的文件。

(3) 静态网页的内容相对稳定，因此容易被搜索引擎检索、访问速度比较快。

(4) 静态网页没有数据库的支持，在网站制作和维护方面工作量较大，因此当网站信息量很大时完全依靠静态网页制作方式比较困难。

(5) 静态网页的交互性差，在功能方面有较大的限制。为了不断更新网页内容，网站管理者必须不断地重复制作 HTML 文件，随着网站内容和信息量的日益增长，维护工作将变得十分复杂。

1.1.3　动态网站

什么是动态网站呢？所谓“动”，并不是指网页上的 GIF 等动画图片，而是指用户与网站的交互性和互动性。动态网站一般应满足以下特征。

1. 交互性

动态网站中的网页会根据用户的要求和选择而作出改变和响应。网站管理员只需要掌握计算机基本操作方法，就可以方便、及时地更新网站内容；浏览网站的用户可以在网站中进行查询和留言等操作。可见，动态网站技术大大增加了客户与网站的交互性。

2. 通过数据库进行架构

在动态网站中，网络管理员除了要设计网页视觉效果，还要设计数据库和程序代码来使网站具有更多自动的和高级的功能。例如，购物网站中含有大量的商品种类和数量信息，为了方便查找，就应搭建数据库平台在网页上实现自动搜索。现在广泛使用的网上交易系统、在线采购系统以及商务交流系统等都是由数据库提供技术支持的。

3. 在服务器端运行，方便更新

在服务器端运行的程序、网页、组件，会随不同客户、不同要求返回不同的页面，网站管理员无须手动更新网页文档，可以大大节省网站管理的工作量，如图 1-2 所示。

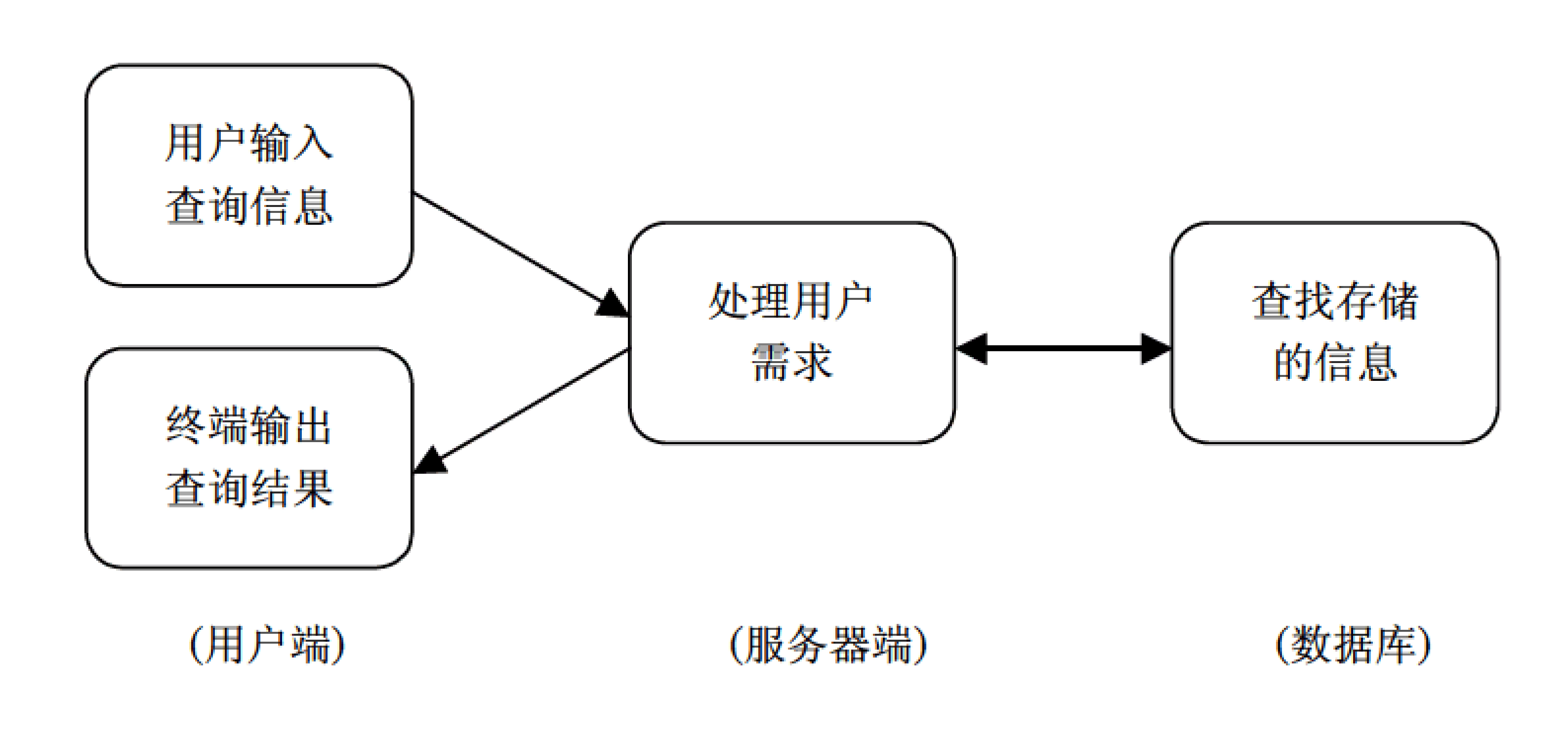


图 1-2　动态网站模型

由上述特征可以看出，静态网站和动态网站的主要区别在于：静态网站的内容是在用户发出请求之前就预先生成的，而动态网站的内容则是在用户发出请求之后才产生的。

动态网站在发出请求之后生成内容有两个明显的优点。

- 首先，服务器端可以根据用户提交的请求以及请求中的信息来生成页面的内容。如，在一个电子商务网站提交用户名和密码，那么用户看到的下一个页面就是动态生成的，它包含了用户的私有账户信息。
- 其次，服务器端可以根据最新的可用信息来设置它所生成的页面内容。如显示网站的当前在线用户数。在线用户数是实时信息，是在 Web 服务器接收用户请求时获取的。

静态网站和动态网站各有特点，搭建网站采用动态还是静态技术主要取决于网站的功能需求和内容的多少，如果网站功能比较简单，内容更新量不是很大，那么采用静态网站的方式会更简单，反之，就要采用动态网站技术来实现。

说明：

静态网站可以使用 Frontpage 或 Dreamweaver 等网页编辑工具来建立，而动态网站则需要使用服务器端网页技术，如通过本书介绍的 ASP.NET 来搭建。

1.2　ASP.NET 与 VWD 2010

ASP.NET 是由 Microsoft 公司提出的动态网站设计技术与程序框架，它带给人们的是全新的技术，和由此产生的开发效率的提高，网站性能的提升。使用 ASP.NET 提供的内置服务器控件或第三方生成的控件，可以创建既复杂又灵活的用户界面，大幅度减少动态网站的开发工作。目前，ASP.NET 作为 Windows 平台上流行的网站开发工具，能够提供各种方便的 Web 开发模型，利用这些模型用户能够快速地开发出动态网站所需的各种复杂功能。

1.2.1　ASP.NET 的发展史

自从.NET Framework 1.0 在 2002 年初首次发布以来，Microsoft 花了大量精力和时间来开发 ASP.NET，它是.NET Framework 的一部分，可以用来构建 Web 应用程序。下面先来看一下 ASP.NET 的发展历史。

早期的 Web 程序开发是十分繁琐的事情，一个简单的动态页面需要编写大量的代码(一般用 C 语言)才能完成。

1996 年，Microsoft 推出了 ASP(Active Server Page，活动服务器页面，现在人们常称之为传统 ASP)1.0 版。它允许采用 VBScript/JavaScript 这些简单的脚本语言编写代码，允许将代码直接嵌入 HTML 中，从而使得设计动态 Web 页面的工作变得简单。ASP 能够通过内置的组件，实现强大的功能(如 Cookie)。ASP 最显著的贡献就是推出了 ActiveX Data Objects(ADO)，它使得程序对数据库的操作变得十分简单。

1998 年，微软发布了 ASP 2.0 和 IIS 4.0。与前一版相比，2.0 版最大的改进是外部的组件需要初始化。用户能够利用 ASP 2.0 和 IIS 4.0 构建各种 ASP 应用，而且每个组件有了自己单独的内存空间，方便进行事务处理。随后，微软在 Windows 2000 Server 系统中提供了 IIS 5.0 和 ASP 3.0。此次升级，最主要的改变就是把很多事情交给 COM+来做，效率比以前的版本更高，而且更稳定。

ASP.NET 是 Microsoft 公司于 2002 年推出的新一代体系结构 Microsoft .NET 的一部分，用来在服务器端构建功能强大的 Web 应用。ASP.NET 1.0 在结构上与传统的 ASP 版本截然不同，它几乎完全是基于组件和模块化的。

说明：

ASP.NET 提出了代码隐藏类(CodeBehind)的概念，把逻辑代码(.aspx.cs)和表现页面(.aspx)分离开来，使用户很容易使用后台代码来控制页面的逻辑功能。

2003 年，Microsoft 公司发布了 Visual Studio .NET 2003，提供了在 Windows 操作系统下开发各类基于.NET 框架的全新的应用程序开发平台(称为.NET 1.1)。

2005 年 11 月，Microsoft 发布了 Visual Studio 2005 和 ASP.NET 2.0。它修正了以前版本中的一些 Bug，并在移动应用程序开发、代码安全以及对 Oracle 数据库和 ODBC 的支持等方面都做了很多改进。

尽管 Visual Studio 2005 和 ASP.NET 2.0 的功能已经很丰富了，但 Microsoft 仍旧努力向 2007 年 11 月发布的 Visual Studio 2008 和 ASP.NET 3.5 中添加了一系列很酷的新功能。主要的新功能包括 LINQ 以及 AJAX 框架整合。

2008 年 8 月，Microsoft 又发布了用于 Visual Studio 和.NET Framework 的 Service Pack 1，其中引入了一些重要的新功能，如 ADO.NET Entity Framework 和动态数据。

目前的最新版本是 Visual Studio 2010(通常读作 twenty-ten)和 ASP.NET 4.0，它是在已成功发行的 Visual Studio 2008 和 ASP.NET 3.5 的基础之上构建的，保留了其中很多令人喜爱的功能，还增加了一些其他领域的新功能和工具。

1.2.2　ASP.NET 的工作原理

ASP.NET 不只是一个运行库宿主，它是使用托管代码开发网站和通过 Internet 分布的对象的完整结构。Web 窗体和 XML Web 服务都将 IIS 和 ASP.NET 用做应用程序的发布机制。

ASP.NET 的工作原理如图 1-3 所示。

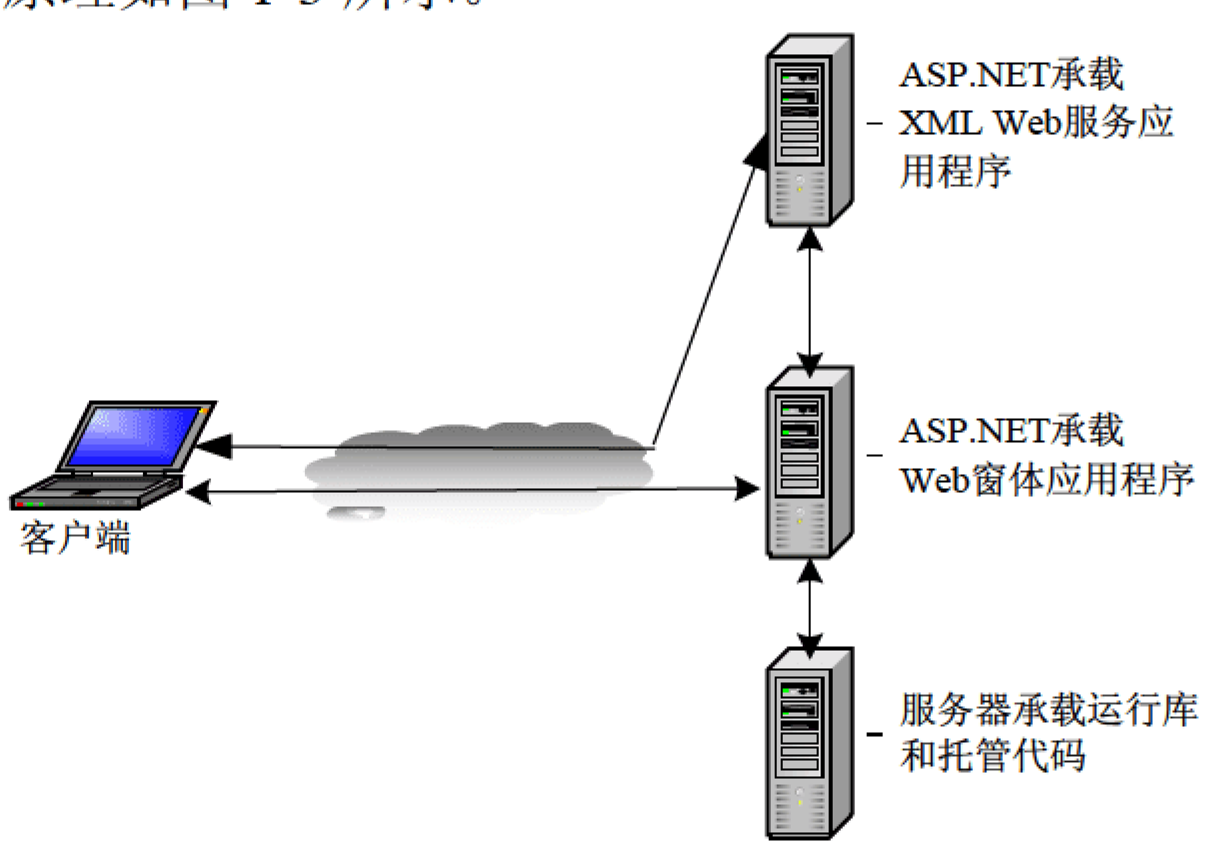


图 1-3　ASP.NET 工作原理

当在浏览器中输入某网站的域名或 IP 地址并按下 Enter 键时，浏览器就会向服务器发送一个请求，这一过程是通过 HTTP(HyperText Transfer Protocol，超文本传输协议)完成的。当服务器是活动状态并且请求有效时，服务器就会接受请求，处理请求，然后将响应发回到客户机浏览器上。

与传统的 ASP 技术相比，ASP.NET 支持.NET Framework 的任何语言开发 Web 窗体页，而且代码不再需要与 HTML 文本共享同一个文件(当然也可以继续共享同一个文件)。Web 窗体页用本机语言执行，这是因为与所有其他托管应用程序一样，它们充分利用运行库。

ASP.NET 页面的处理过程如下。

- 用户通过客户端浏览器请求页面，页面第一次运行，执行初步处理。
- 执行的结果以标记的形式呈现给浏览器，浏览器对标记进行解释并显示给用户。
- 用户输入信息或者进行选择或者单击按钮等进行交互。
- 页面发送回服务器，在 ASP.NET 中称此为“回发”。
- Web 服务器接到回发请求，在此运行该页，并且使用用户输入或选择的信息。
- 服务器将运行后的页面以 HTML 或 XHTML 标记的形式发送到客户端浏览器。

1.2.3　VWD 2010

进行 ASP.NET 开发可以使用 Visual Basic.NET 或者 C#，这两种语言都是.NET 环境下的程序设计语言，但并不是必须使用.NET 集成开发环境才能进行 ASP.NET Web 程序设计。因为 ASP.NET 文件实际上是一个纯文本的文件，编译工作是在用户向服务器第一次发出对该文件的 HTTP 请求时由 Web 服务器进行的，并不是由 VS 完成的。从理论上讲，用记事本或其他文本编辑器就可以编写 ASP.NET Web 应用程序，但大多数开发人员还是希望安装 Microsoft Visual Web Developer 2010(VWD)。VWD 是专门为构建 ASP.NET Web 站点而开发的，其中包含了大量有助于快速创建复杂 ASP.NET Web 应用程序的工具。

Visual Web Developer 有两个版本：一个是独立而免费的版本，称为 Microsoft Visual Web Developer 2010 Express；还有一个版本是作为较大的开发套件 Visual Studio 2010 的一部分，它有不同的版本可用，且各个版本的价格各不相同。虽然 VWD 的 Express 版本是免费的，但是它包含了创建复杂且功能丰富的 Web 应用程序所需的所有功能和工具。本书中的所有示例都可以用该版本构建出来。

1. 获取 Visual Web Developer 2010

可以从 Microsoft 站点 http://www.microsoft.com/express/上下载 VWD 的免费版本。在 Express 的主页上，依次单击 Download 链接，直到打开提供了下载 Express 产品的下载页面，其中包括 Visual Web Developer 2010 Express 版本。在这个页面上可以以 Web 安装方式下载 Visual Web Developer 2010 Express 版本，这里只下载安装程序，文件的其余部分将在安装过程中下载。

也还可以从 www.microsoft.com/web 和 www.asp.net/vwd/站点上下载 Microsoft Web Platform Installer(WPI)应用程序，其中就包含了 VWD。除了 VWD 以外，这个工具还便于访问其他许多与 Web 开发相关的工具和程序。通过使用 WPI 这个优秀的工具，可以同时获得大量与 Web 开发相关的程序和工具。

说明：

在本书中，以 C#作为编程语言，以 VWD 2010 作为开发环境，进行 ASP.NET 动态网站开发，因此需要把它安装到开发机器上。

2. 安装 VWD Express 版本

Visual Web Developer 的安装很简单，只是过程有点长。根据所选的安装方法、计算机配置和 Internet 连接速度，安装 VWD 可能需要半个小时到一个小时，甚至更长时间。

安装 Visual Studio 2010 的完整版本与之类似，只是中间步骤可能略有不同。不管安装 VWD 的哪个版本，都要安装 SQL Server 2008，本书的很多示例都会用到这个组件。如果安装的是 Visual Studio 2010 的完整版，那么在安装过程中会看到要安装的功能列表中包括安装 SQL Server 的选项。如果安装 VWD Express 版本，“安装选项”对话框中就会出现选择 SQL Server 的选项。

安装步骤如下：

(1) 运行从 Microsoft Web 站点上下载的文件。或者从安装光盘上启动安装过程。将开始下载 Web 平台安装程序文件。

(2) 下载完以后，打开“Web 平台安装程序 1.0”，单击“安装”按钮，出现软件的许可条款，如图 1-4 所示。在该列表中，包括 VWD 2010 依赖的所有应用程序和组件。如果没有看到 SQL Server 选项，则表示已经安装过了。

(3) 阅读并接受许可条款，单击“我接受”按钮将开始下载安装，下载和安装的过程与需要安装的组件有关。

(4) 完成了应用程序的安装后，可能会出现一个对话框，要求重启计算机。重启后，VWD 就可以使用了。



图 1-4　软件许可条款

3. VWD 2010 提供的功能

VWD 2010 提供了如下功能。

- 网页设计：VWD 2010 内置功能强大的网页编辑器，包含所见即所得的编辑模式和 HTML 编辑模式，以及智能感应功能和验证功能。支持所见即所得的拖动界面，可以创建美观、易用的网站。
- 代码编辑：VWD 2010 提供代码编辑器，使用户可以使用 Visual Basic .NET 或 C# 编写动态网页的代码。代码编辑器包括语法修饰和智能感应功能。
- 调试：提供调试器，帮助用户查找程序中的错误。
- 控件：ASP.NET Web 服务器控件整合了创建网站所需的大部分功能，用户可以快速开发 Web 应用程序。
- 数据访问：支持用户在网页中显示和编辑数据。数据可以位于各种数据存储区中，其中包括数据库或 XML 文件。在很多情况下，用户无须编写任何代码，即可向网页中添加和编辑数据。
- 对文件传输协议(FTP)的内置支持：使用 VWD 2010 的 FTP 功能，可以直接连接到服务器，然后在该服务器上创建和编辑文件。
- 内置 Web 服务器：VWD 2010 包含了一个内置的 Web 服务器，方便开发人员创建和调试 ASP.NET Web 应用程序。因此，用户不需要再安装和配置 IIS 服务器，就可以开发 ASP.NET Web 应用程序。
- 微软 AJAX：微软 AJAX 的一个重要特性是，它与其他客户端架构(包括 jQuery)具有很好的互操作性。除了实现无闪烁页面的控件之外，微软 AJAX 还提供了更多的服务器控件来创建交互式的且有响应的用户界面。
- jQuery 1.4：jQuery 库的主要关注点一直是简化访问 Web 页面元素的方法、帮助处理客户端事件、提供视觉效果(如动画)支持，以及在应用程序中 Ajax 的使用变得更加简单。VWD 2010 包含了目前最新的稳定版本 jQuery 1.4。使用 ASP.NET Web 站点模板创建的任意新 Web 站点都包含一个 Scripts 文件夹，其中已经包含了必要的 jQuery 文件。
- MVC2.0：ASP.NET MVC 模式是一种表现模式。它将 Web 应用程序分成 3 个主要组件，即：模型(Model)、视图(View)和控制器(Controller)。在 ASP.NET MVC 中，“请求——处理——响应”的模型变得更加简单。View 和 Controller 之间不再有强耦合，而且页面没有复杂的生命周期。
- 多显示器支持：比如将代码编辑器放置在主显示器中，将输出窗口、类图窗口和代码定义窗口等提供辅助信息的窗口放置在副显示器中，这样就可以在主窗口中编辑代码，同时可以在有需要的时候，可以及时地从辅助窗口中得到一些有用的辅助信息。

1.3　使用 VWD 2010 开发 Web 应用程序

现在已经安装了 VWD，接下来就可以启动并使用它来创建 ASP.NET 应用程序了。本节将介绍 VWD 2010 的常用窗口和基本界面，以及如何使用 VWD 2010 创建 ASP.NET Web

应用程序。

1.3.1　启动 VWD 2010

单击 Windows 的“开始”菜单，选择“所有程序”| Microsoft Visual Studio 2010 Express | Microsoft Visual Web Developer 2010 Express 命令，即可启动 VWD 2010，如图 1-5 所示。

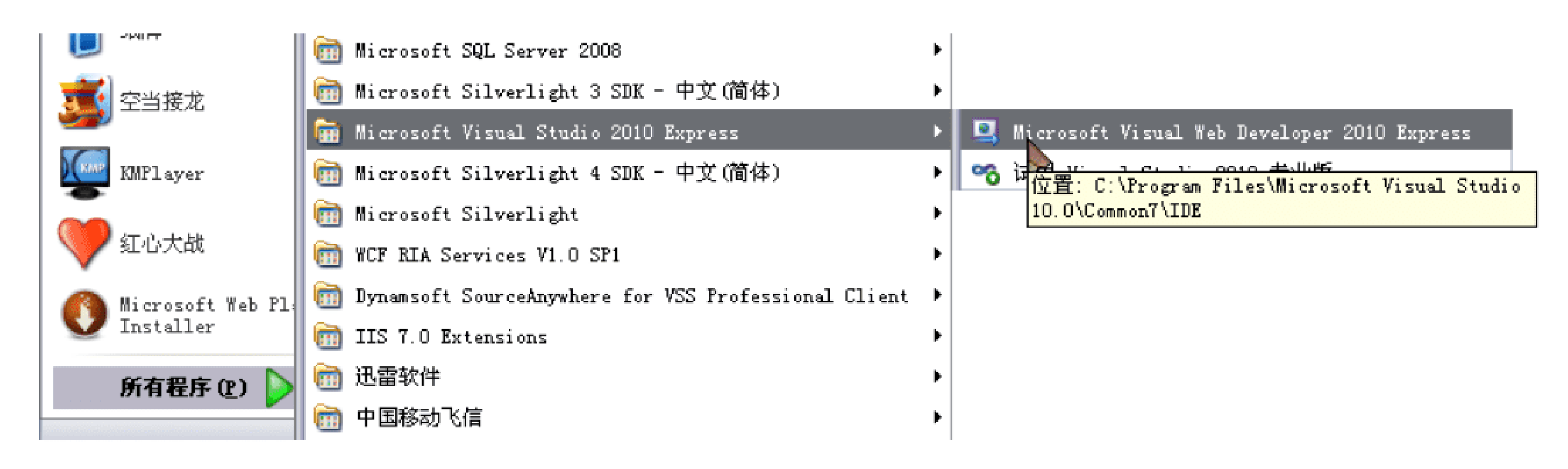


图 1-5　通过开始菜单启动 VWD 2010

首先出现的是软件版本界面，几秒钟之后，软件版本界面消失，出现“起始页”界面，如图 1-6 所示。

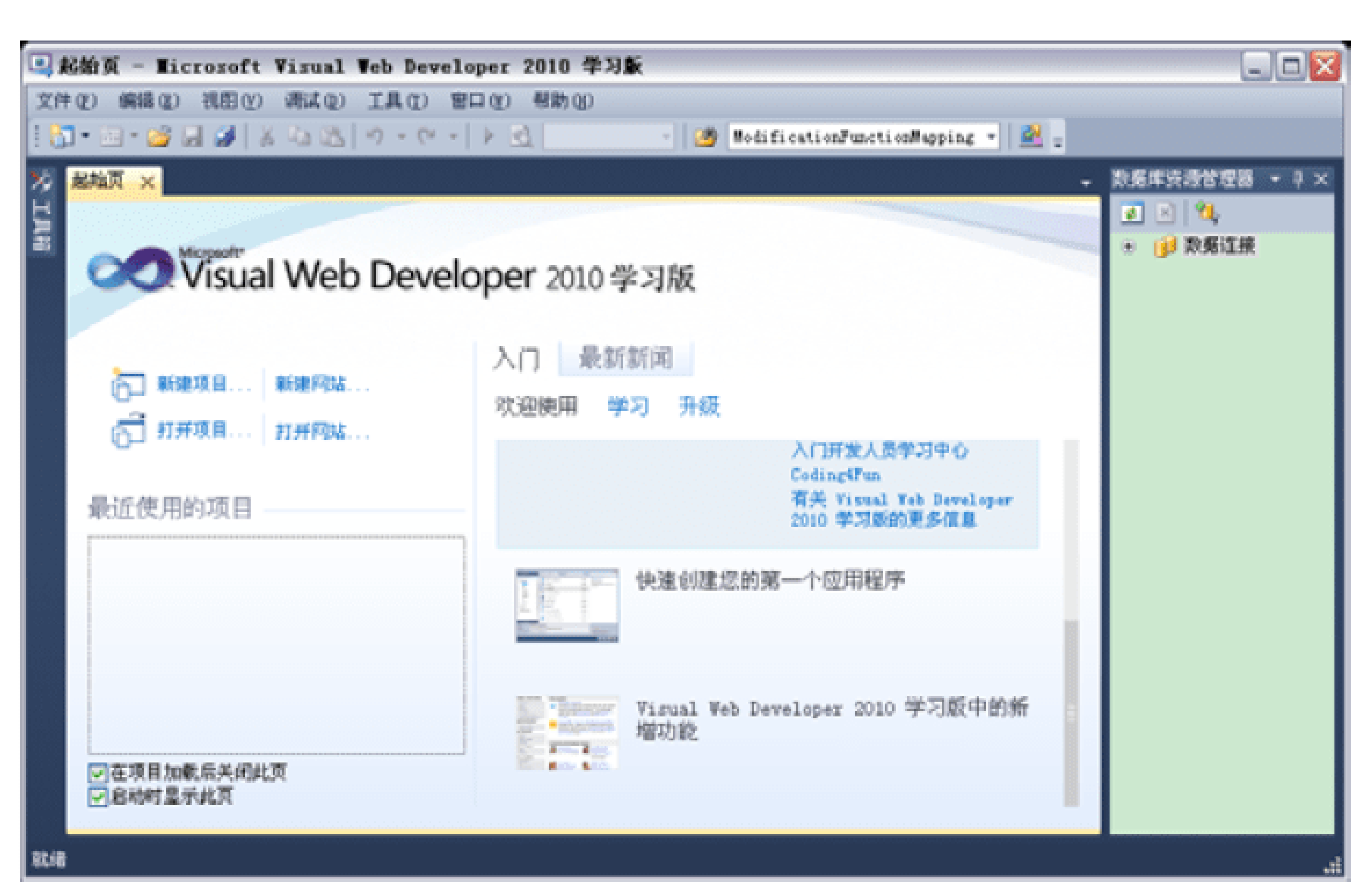


图 1-6　“起始页”界面

起始页包括“最近使用的项目”和联机资源以及新建和打开项目的快捷操作。为了介绍 VWD 2010 的操作环境，先新建一个网站。

在“起始页”中单击“新建网站”链接，或者选择“文件”|“新建网站”命令，打开“新建网站”对话框，如图 1-7 所示。

该对话框显示了已经安装的模板，这里选择“ASP.NET 网站”模板，在“Web 位置”下拉列表中选择“文件系统”选项，然后在后面的文本框中输入存储位置，或者单击“浏览”按钮选择一个新位置，单击“确定”按钮即可创建一个 ASP.NET 网站。

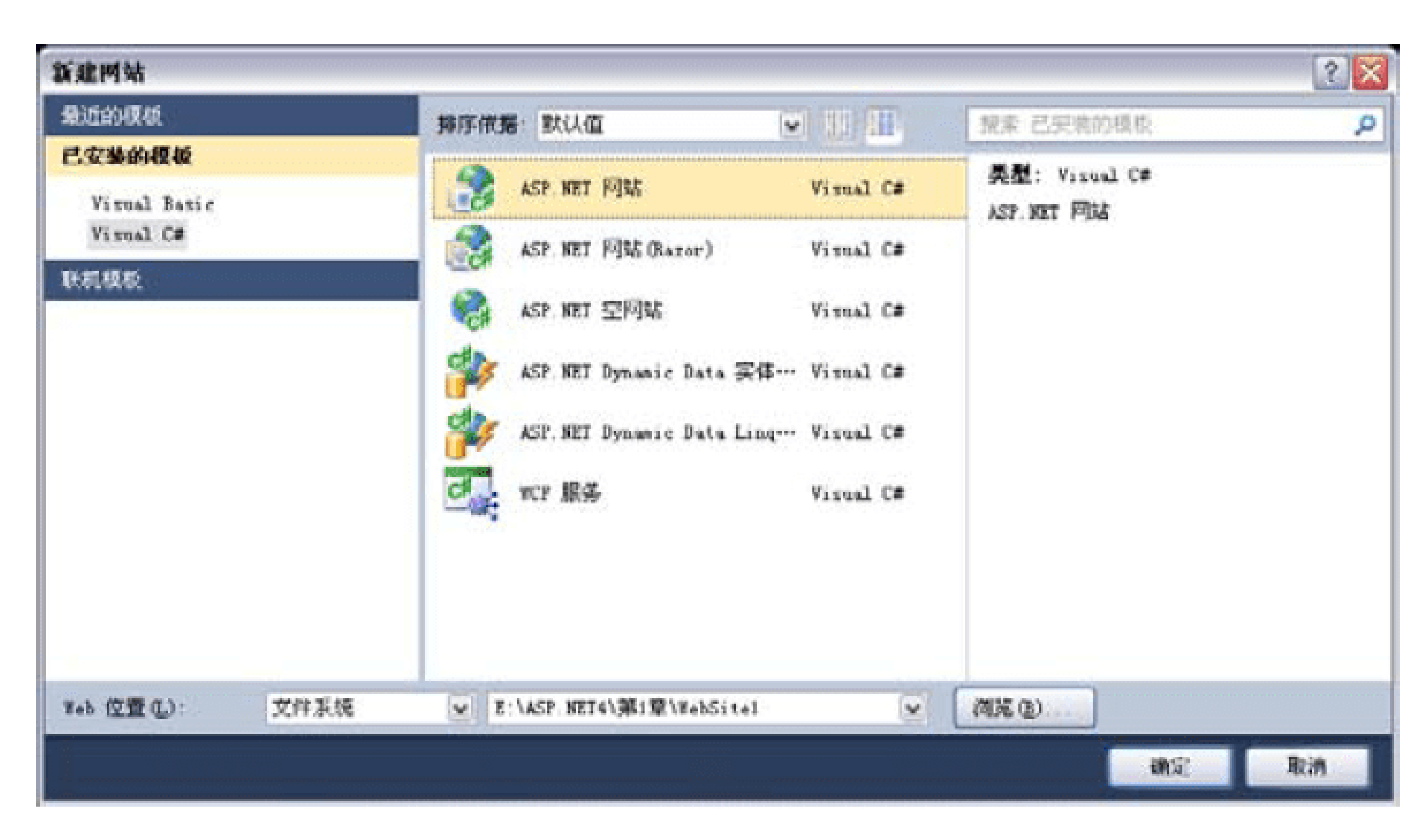


图 1-7　“新建网站”对话框

通过此模板新建的网站是一个基于母版页 Site.master 的简易站点。其中包括名为 Default.aspx 的标准页面，一个 web.config 文件，一组包括注册、登录、修改密码等功能的通用页面，一个关于页面 About.aspx，以及一个空的 App_Data 文件夹，如图 1-8 所示。

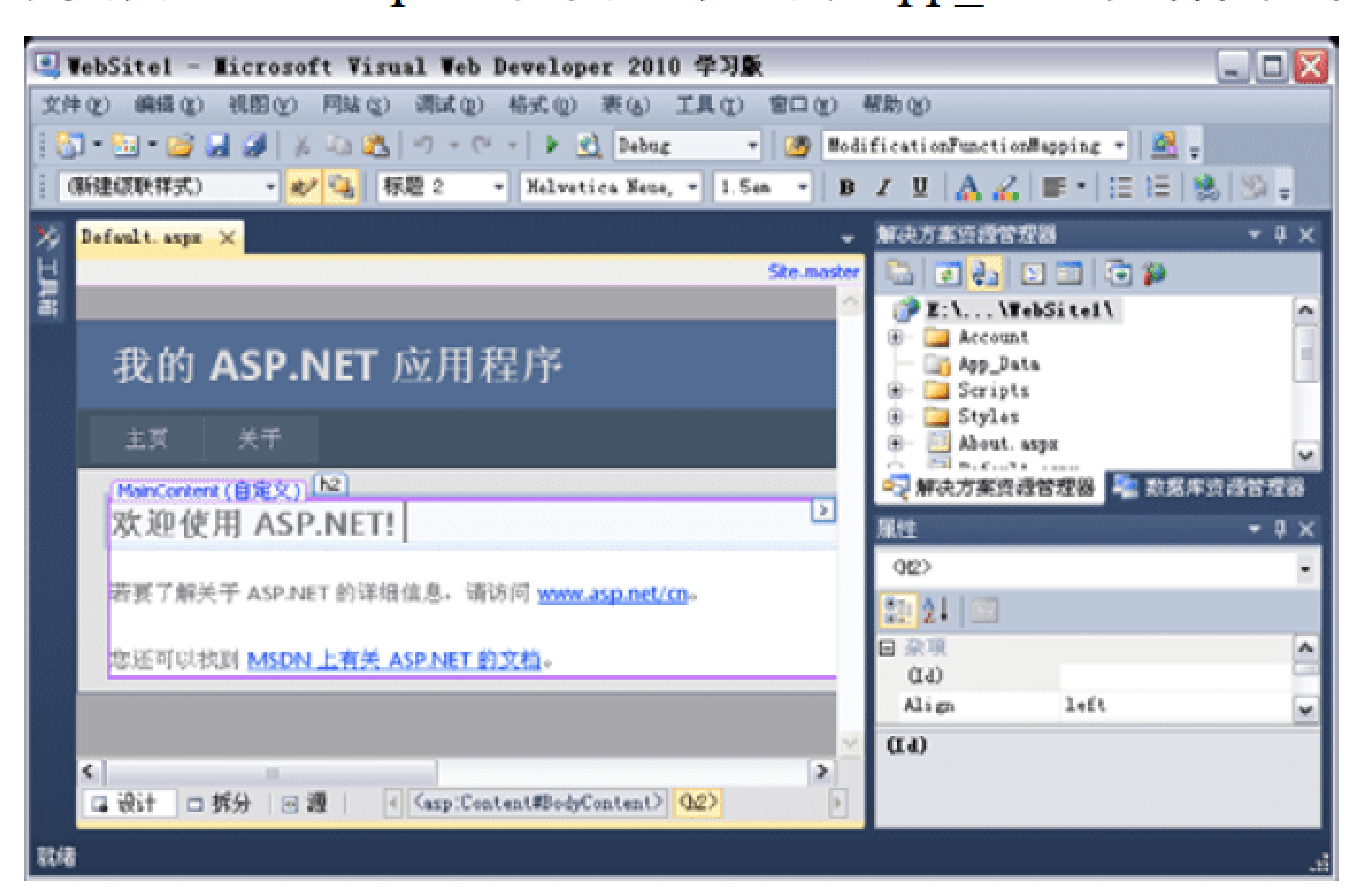


图 1-8　VWD 2010 主开发界面

说明：

使用“ASP.NET 网站”模板新建的站点包括了注册登录以及后台数据库的相关功能，不适合初学者，所以本书后面将介绍的站点大都基于“ASP.NET 空网站”模板创建。

VWD 2010 的主界面包括标题栏、菜单栏、工具栏、工具箱、解决方案资源管理器、数据库资源管理器、属性窗口和文档窗口等。

1. 菜单栏

开发界面的最上方是标题栏，标题栏的下面就是菜单栏，包括“文件”、“编辑”、“视图”、“网站”、“调试”、“格式”、“工具”、“窗口”和“帮助”9 个主菜单。根据执行的具体任务，主菜单也会有很大的变化，因此，在使用应用程序的过程中就会发现某些菜单有时出现，有时消失。

2. 工具栏

菜单栏的下面就是工具栏，利用不同的工具栏，可以快速地访问 VWD 中的大部分常用功能。图 1-9 中只显示了 2 个工具栏(标准和格式设置)，如果要打开或关闭某个工具栏，可以右击现有的工具栏，或者选择“视图”|“工具栏”菜单，从弹出的子菜单中选择相应的菜单项即可，如图 1-9 所示。

3. 工具箱

默认情况下，在主窗口的左侧，可以看到折叠的工具箱选项卡，将鼠标指针移动到该选项卡上悬停几秒，工具箱就会展开，如图 1-10 所示。

与菜单栏和工具栏一样，在执行不同的任务时，工具箱也可能会变化，以显示相关的控件。可以简单地通过拖动鼠标的方式将工具箱中的控件放到页面中的合适位置。工具箱中的控件包含多个分类，用户可以根据需要展开或折叠某个分类，以便找到需要的控件。本书第 3 章将重点介绍工具箱中各种控件的使用。

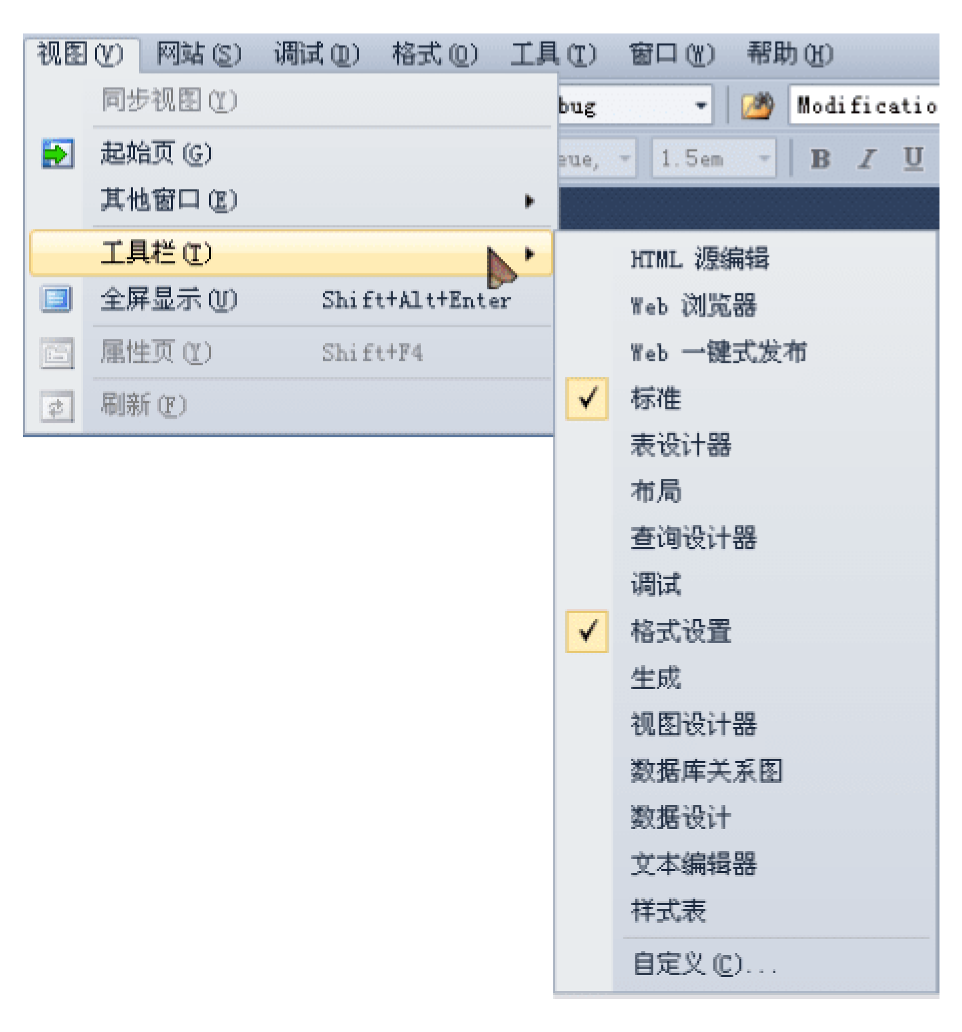


图 1-9　“视图”|“工具栏”子菜单

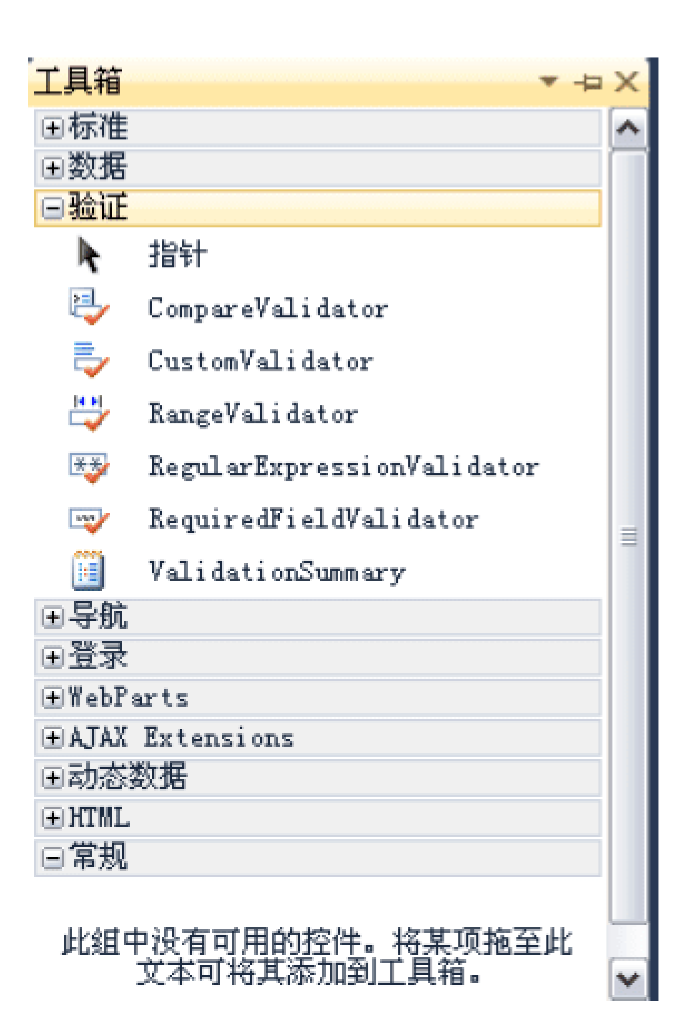


图 1-10　工具箱

提示：

如果在主窗口左侧找不到工具箱，可以使用 Ctrl+Alt+X 组合键或者选择“视图”|“其他窗口”|“工具箱”命令来打开它。

4. 解决方案资源管理器

窗口的右上角是“解决方案资源管理器”窗口，如图 1-11 所示。在“解决方案资源管理器”窗口中，文件被分门别类地存储在不同的文件夹中，可以通过该窗口向站点中添加新的文件夹和文件，也可以从项目中删除文件或更改文件或文件名。解决方案资源管理器的大部分功能都集中在它的右键菜单中。

在“解决方案资源管理器”窗口的位置还有一个“数据库资源管理器”窗口，通过该窗口，可以使用数据库，它提供了创建新数据库和打开现有数据库、向数据库中添加新的表和查询工具。

5. 属性窗口

“属性”窗口位于界面的右下角，如图 1-12 所示。通过该窗口可以查看和编辑项目、文件、控件和页面本身的属性以及其他更多内容。

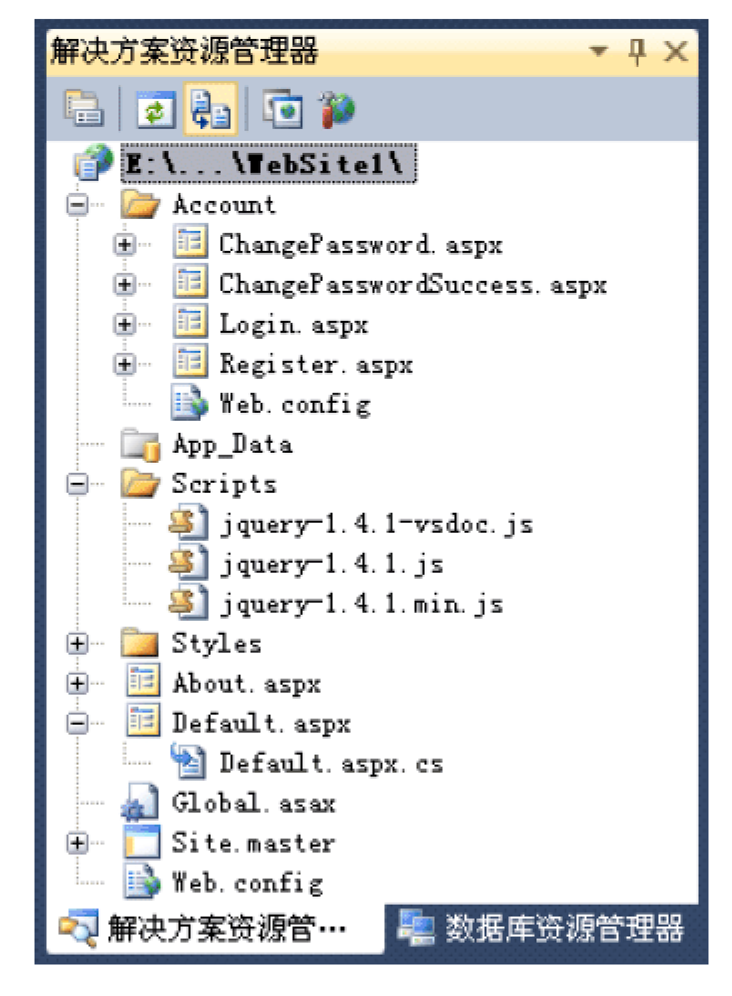


图 1-11　“解决方案资源管理器”窗口

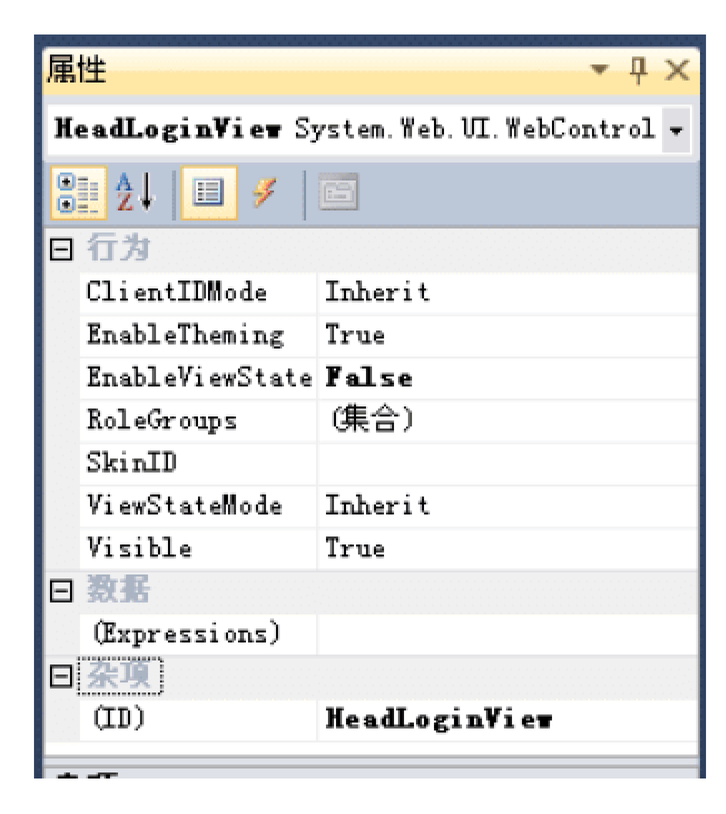


图 1-12　“属性”窗口

6. 文档窗口

文档窗口是界面的主要区域，大部分动作都是在这里发生的。在文档窗口的下面，有 3 个视图按钮：“设计”、“拆分”和“源”。在操作含有标记的文件(如 ASPX 和 HTML 文件)时，这些按钮会自动出现。单击“设计”按钮可以打开页面的设计视图窗口，在这里可以看到页面在浏览器中的效果；单击“拆分”按钮，窗口将一分为二，同时打开设计和源视图；单击“源”按钮打开源视图，在此可以看到页面的源代码文件。

技巧：

默认情况下，文档窗口是一个带选项卡的窗口，各文件之间通过选项卡分隔，文件名在窗口的顶部。如果选项卡上的文件名带有“*”，则说明该文件内容修改过但还没有保存。

可视化的 VWD 大大增加了文档窗口中文本显示方式的灵活性。读者可以修改一些属性，如字体大小、字体颜色，甚至是文本的背景色。选择“工具”|“选项”命令，可以打开“选项”对话框，如图 1-13 所示。

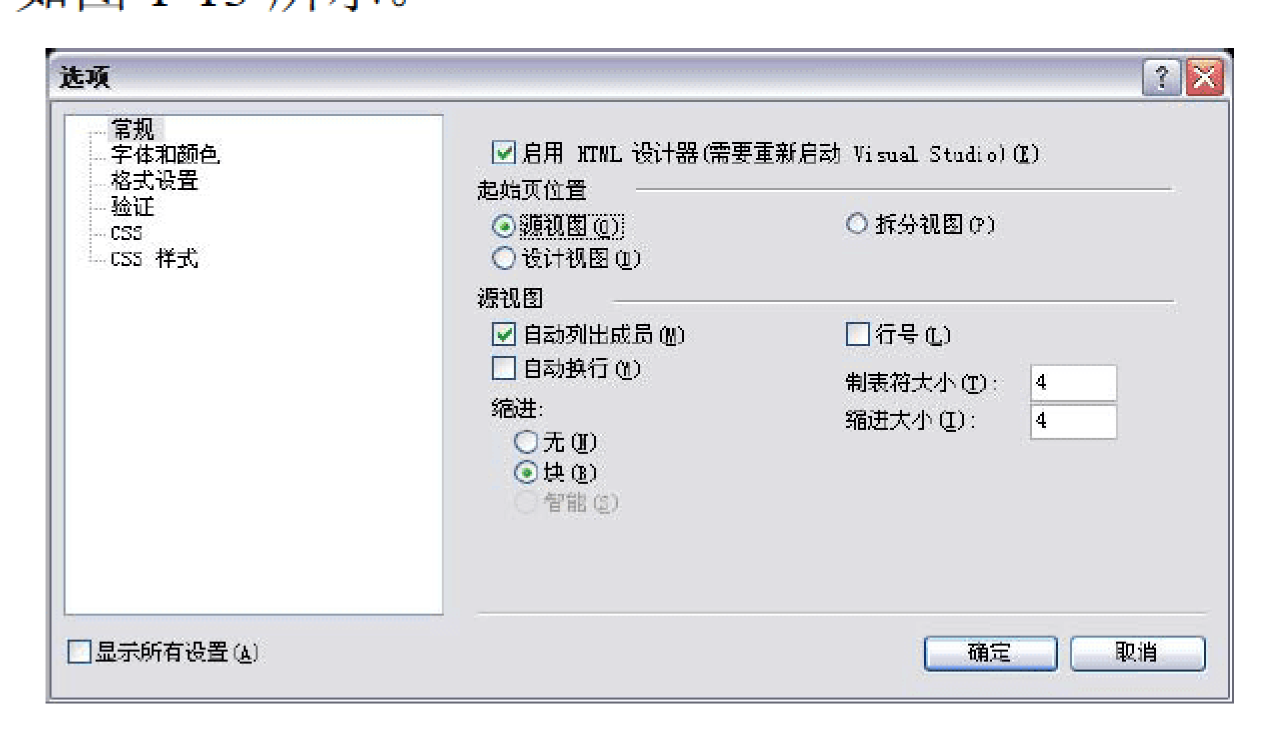


图 1-13　“选项”对话框

在“常规”选项页中可以设置“起始页位置”、“源视图”和制表符大小等。在“字体和颜色”选项页中可以设置文字的字体、大小和颜色等。

如果选中下方的“显示所有设置”复选框，还可以设置“环境”、“项目和解决方案”“调试”等选项，如图 1-14 所示。

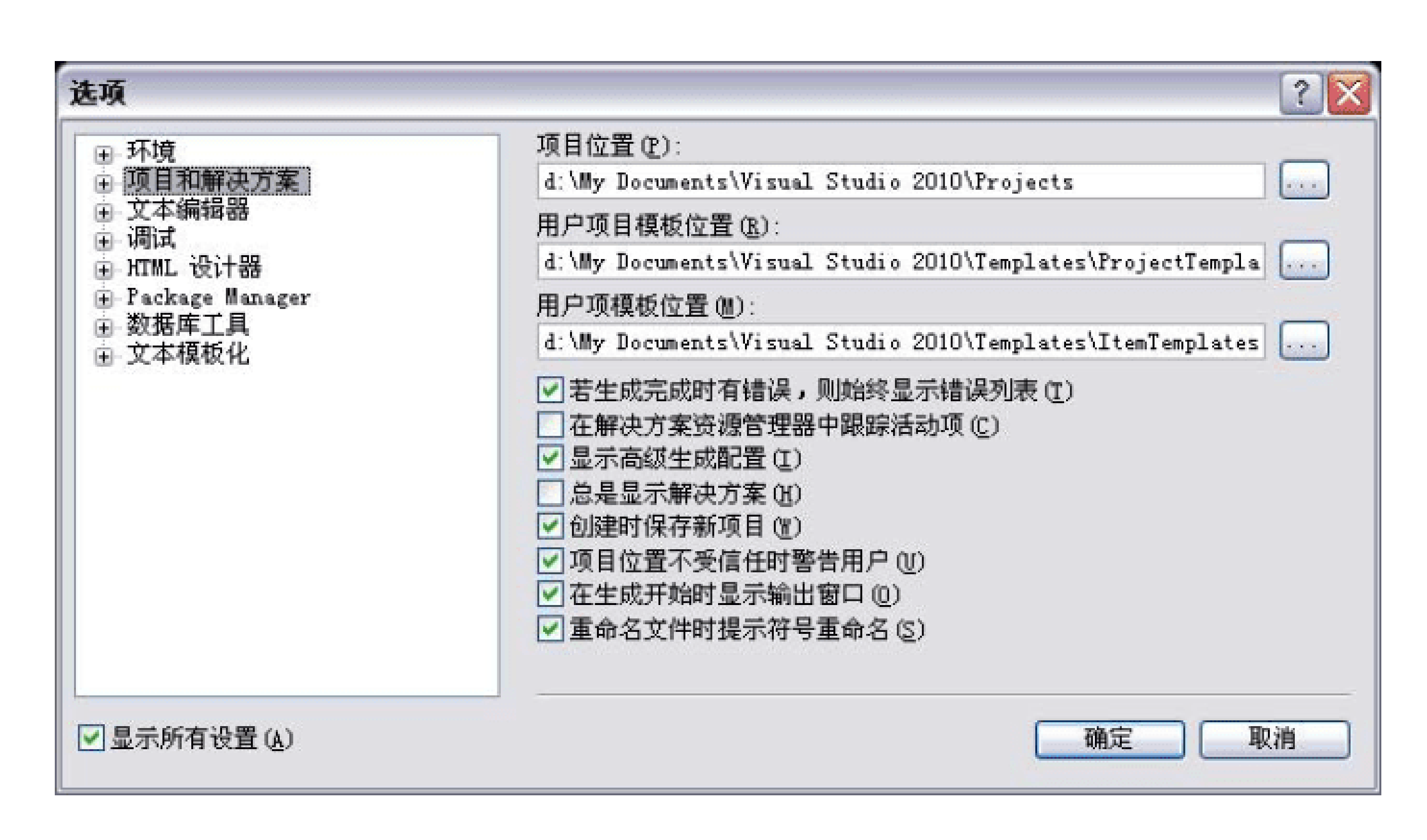


图 1-14　显示所有设置的“选项”对话框

7. 其他窗口

除了上面介绍的窗口之外，VWD 2010 还有很多其他的窗口，包括“输出”窗口、“错误列表”窗口、“书签”窗口和“查询结果”窗口等。这些窗口都可以通过“视图”菜单下面的相应命令打开。

1.3.2　第一个 ASP.NET 应用程序

本节将通过 VWD 2010 创建第一个 ASP.NET Web 应用程序。通过这个 Web 应用程序，读者将学会如何使用 VWD 2010 创建 Web 应用程序，并对在浏览器中浏览 ASP.NET 页面时后台的工作有一个很好的了解。

1. 新建网站

例 1-1：通过 VWD 2010 新建网站，了解通过 VWD 2010 开发 Web 应用程序的步骤。

(1) 通过“开始”菜单启动 VWD 2010，选择“文件”|“新建网站”命令，打开“新建网站”对话框。

(2) 选择“ASP.NET 空网站”模板，在“Web 位置”下拉列表中选择“文件系统”选项，然后在后面的文本框中输入存储位置，存储位置的最后是网站名“Chapter1”，单击“确定”按钮即可创建一个空网站 Chapter1。通常情况下，系统会为新建网站创建一个新的子目录。

新建的空网站只有一个名为 web.config 的配置文件。下面将添加一个 Web 窗体。

(3) 在“解决方案资源管理器”窗口中右击解决方案，从弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 1-15 所示。

选择“Web 窗体”模板，默认文件名为 Default.aspx，单击“添加”按钮即可。

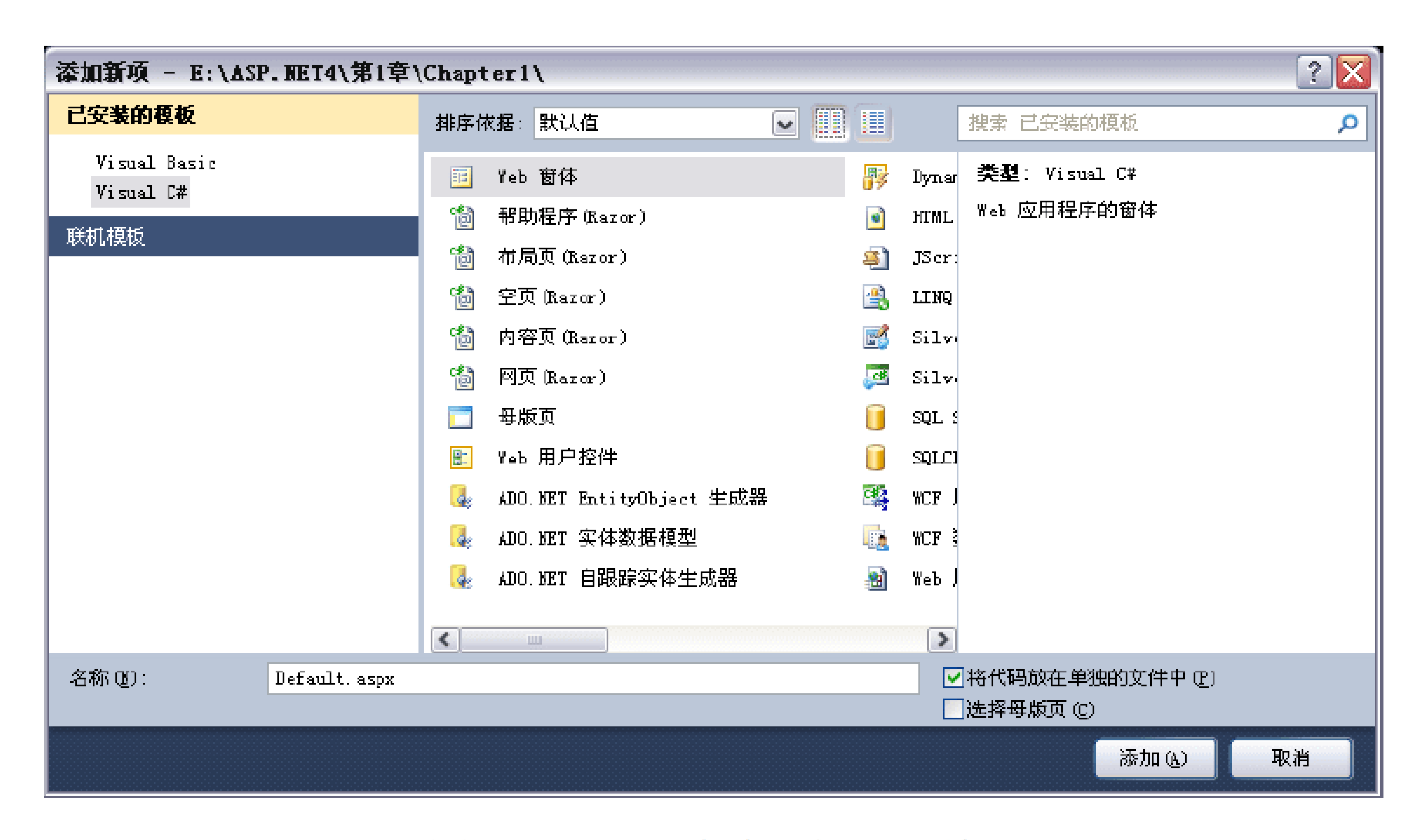


图 1-15　“添加新项”对话框

说明：

一个完整的 ASP.NET 页面文档通常是由指令、文档类型声明、代码声明、服务器代码、文本和 XHTML 标记等部分组成。现在，读者不必深入了解这些，下一节将进行详细介绍。

(4) 在 Defaute.aspx 页面的“源”视图窗口中，修改页面的代码如下所示：

```
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w1.org/1999/xhtml">
    <head runat="server">
        <title>我的第一个 ASP.NET 页面</title>
    </head>
    <body>
        <h2>从今天开始学习 ASP.NET 4 </h2>
        <form id="form1" runat="server">
        <div>
        欢迎使用 VWD 2010，现在时间是：<% =System.DateTime.Now %>
        </div>
        </form>
    </body>
    </html>
```

读者可能还不完全熟悉所有的代码，但是不难猜出它是用来输出当前日期和时间的。

(5) 选择“调试”|“启动调试”命令，或者按 F5 键，或单击工具栏中的▶按钮，将编译并生成网站，同时启动调试。

(6) 如果代码输入正常，主窗口下方的“输出”窗口中将出现生成成功的信息，如图 1-16 所示。如果有语法错误，则在“错误列表”中将逐一列出所有错误，双击相应的错误将跳转到相应的代码行。

(7) 此时将弹出“未启用调试”对话框，如图 1-17 所示，如果选择“修改 Web.config 文件以启用调试”单选按钮，则以后启动该工程时将不再弹出该对话框，而是默认启动调试；如果选择“不进行调试直接运行”单选按钮，则不启动调试，等同于用户按 Ctrl+F5 组合键。

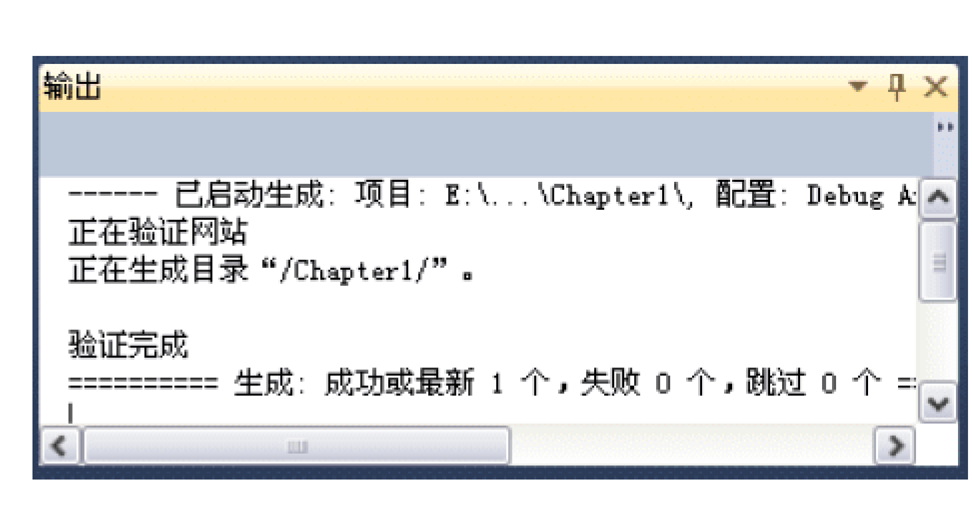


图 1-16 “输出”窗口



图 1-17 “未启用调试”对话框

(8) 单击“确定”按钮后，将自动启动默认的 Web 浏览器，同时打开该页面，如图 1-18 所示。

(9) 此时，在 Windows 的任务栏中会出现一个带屏幕提示的小图标，这个图标属于 ASP.NET Development Server。该 Web 服务器由 VWD 自动启动，以响应对页面的请求。双击该图标将打开如图 1-19 所示的详细信息。



图 1-18 页面运行效果



图 1-19 ASP.NET Development Server

2. 工作原理

虽然本例中创建的 Web 站点非常简单，但是让 Default.aspx 页面显示在浏览器中的过程却没有那么简单。ASP.NET 页面(根据它的扩展名，也称为 ASPX 页面)本身并不能做太多的事。在浏览器能够显示它之前，需要一个 Web 服务器对它进行处理。这就是 VWD 启动内置的 ASP.NET Development Server 来处理页面请求的原因。接下来，它会启动默认的 Web 浏览器并定向到本例中的 Web 服务器地址：http://localhost:46312/Chapter1/Default.aspx。

注意：

每次启动 Web 服务器时地址中的端口号可能会不同，因为该端口是 VWD 随机选择的。

当在浏览器中请求一个 ASPX 页面时，Web 服务器就会处理该页面，执行它在文件中找到的所有代码，并有效地将 ASP.NET 标记转换为纯 HTML，然后发送回客户端浏览器。

要查看最终的 HTML 与原始的 ASPX 页面的区别，可以在浏览器中打开该页面的源代码。选择“查看”|“源文件”命令，将会打开一个默认的文本编辑器，用于显示该页面的HTML 代码。其中的大部分 HTML 与原始的 ASPX 页面相似。然而，如果看一下显示欢迎消息和当前日期与时间的那一行代码，就会注意到它们有很大的区别。除了显示日期和时间的代码行不同以外，还多了一个隐藏字段_VIEWSTATE，如下所示，该字段表示视图状态，第 2 章将详细介绍视图状态的使用。

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head><title>
     我的第一个 ASP.NET 页面
    </title></head>
    <body>
        <h2>从今天开始学习 ASP.NET 4</h2>
        <form method="post" action="Default.aspx" id="form1">
    <div class="aspNetHidden">
    <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUJOTU4MjMyMzI1ZGSyc+01wimxfCCOZ5aiaicW5fnSy47rkrlNaHlTC1PlJQ==" />
    </div>
        <div>
        欢迎使用 VWD 2010，现在时间是：2012-3-2 20:24:18
        </div>
        </form>
    </body>
    </html>
```

1.3.3　ASP.NET 页面文档的结构

一个完整的 ASP.NET 页面文档通常是由指令、文档类型声明、代码声明、服务器代码、文本和 XHTML 标记等部分组成。

1. 指令

ASP.NET 页面通常包含一些指令，允许用户指定页面的属性和配置信息，对页面进行设置，指令指定的设置不会出现在浏览器端。

如在例 1-1 中 Default.aspx 页面的第一行就是 Page 指令，该指令指出使用的语言是 C#，后台代码文件是“Default.aspx.cs”，该页面对应的后台类是_Default。

```
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
```

ASP.NET 的页面包含两个部分：一部分是可视化元素，包括标签、服务器控件以及一些静态文本等；另一部分是页面的程序逻辑，包括事件处理句柄和其他程序代码。ASP.NET

提供两种模式来组织页面元素和代码：一种是单一文件模式；另一种是后台代码模式。

注意：

单一文件模式和后台代码模式功能是一样的，可以在两种模式中使用同样的控件和代码，但要注意使用的方式不同。

- 在单一文件模式下，页面的标签和代码在同一个.aspx 文件中，程序代码包含在<script runat="server"></script>的服务器程序脚本代码块中间，并且代码中间可以实现对一些方法和属性以及其他代码的定义，只要在类文件中可以使用的都可以在此处进行定义。运行时，单一页面被视为继承 Page 类。
- 后台代码模式将可视化元素和程序代码分别放在不同的文件中，如果使用 C#语言，则可视化页面元素为.aspx 文件，程序代码为.cs 文件，根据使用语言的不同，代码后缀也不同，这种模式也被称为代码分离模式。

2. 文档类型声明 DOCTYPE

文档类型声明 DOCTYPE 用于指定文档遵从的 DTD(Document Type Definition，文档类型定义)标准，同时指定了文档的 XHTML 版本，可以和哪些验证工具一起使用等信息，以保证此文档与 Web 标准一致。

例如，例 1-1 中页面的文档类型声明如下：

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

其中，各部分的含义如下。

- DOCTYPE 是 document type(文档类型)的缩写。
- W3C//DTD XHTML 1.0 Transitional 说明此文档符合 W3C 制定的 XHTML 1.0 规范，即声明此文档应该按照 XML 文档规范来配对所有标记。
- xhtml1-transitional.dtd 中的 DTD 是文档类型定义，包含了文档的规则，浏览器根据页面所定义的 DTD 来解释页面内的标识，并将其显示出来。

3. 代码声明

代码声明包含了 ASP.NET 页面的所有应用逻辑和全局变量声明、子例程和函数。页面的代码声明位于<script>…</script>标记中。内联代码位于<%...%>中，如例 1-1 中输出当前时间的代码：<% =System.DateTime.Now %>

4. 服务器代码

大多数 ASP.NET 页面都包含处理页面时在服务器上运行的代码。页面的代码位于 script 标记中，并且该标记包含 runat="server"属性，说明页面运行时，将标记标识为服务器控件，并使其可用于服务器代码。

5. 文本和 XHTML 标记

页面的文本部分用 XHTML 标记来实现，这一部分结构应完全符合 HTML 文件结构。

1.4　本章小结

本章介绍了网站建设相关的基础知识，以及 ASP.NET 4.0 与 VWD 2010 初步介绍。首先，介绍了 HTML、静态网站和动态网站的基本概念。接着介绍了 ASP.NET 的发展史及其工作原理，VWD 2010 的获取与安装。最后，介绍了 VWD 2010 的集成开发环境，以及如何使用 VWD 2010 开发 ASP.NET Web 应用程序。本章内容是全书的基础，下一章将重点介绍 ASP.NET 的内置对象。

1.5　思考和练习

1. HTML 与 XHTML 有何区别。两者有什么关系？
2. 参照本章内容，获取 VWD 2010 并安装。
3. 简述 ASP.NET 的工作原理。
4. 如何设置文档窗口中制表符的大小？
5. 上机操作：新建一个网站，添加一个 ASP.NET 页面，并显示当前时间。

第2章　ASP.NET基础知识

本章主要介绍 ASP.NET 的一些基础知识，学习和掌握这些知识是以后进行 ASP.NET 程序开发的基础和前提。主要包括 ASP.NET 的文件类型和应用程序的目录结构，ASP.NET 的内置对象以及 ASP.NET 的配置文件 web.config 和全局文件 Global.asax。本章是学习 ASP.NET 非常关键的一章，是从认识了解到使用 ASP.NET 的一个关键点。

本章学习目标：

- 了解 ASP.NET 的文件类型
- 掌握 Page 类和窗体页指令
- 使用 ASP.NET 内置对象
- 网页的重定向
- 掌握 Cookie 的使用、设置以及修改的方法
- 视图状态的使用和关闭
- 在 web.config 中创建用户变量
- Global.asax 文件的作用

2.1　ASP.NET 应用程序概述

ASP.NET 应用程序与传统的桌面型应用程序不同。传统的桌面型应用程序是一个独立的 exe 文件，而 ASP.NET 应用程序则总是被分成若干个 Web 页面。这样，用户就可以从不同的入口进入不同的 ASP.NET 应用程序，或者跟随超链接从一个 Web 应用程序导航到另一个 Web 应用程序。

每个 ASP.NET 应用程序都共享一组资源和配置设置。另一个 ASP.NET 应用程序则不能共享这些资源和配置，即使它们位于同一个 Web 服务器上。从技术的角度来讲，每一个 ASP.NET 应用程序都在一个独立的“应用程序域”(application domain)中执行。

如图 2-1 所示是一个 Web 服务器的应用程序域的结构，其中包含了两个独立的 Web 应用程序。

说明：

应用程序域是内存中相互隔离的内存区域，这使得当一个 Web 应用程序出现故障时，并不会影响到另一个 Web 应用程序。

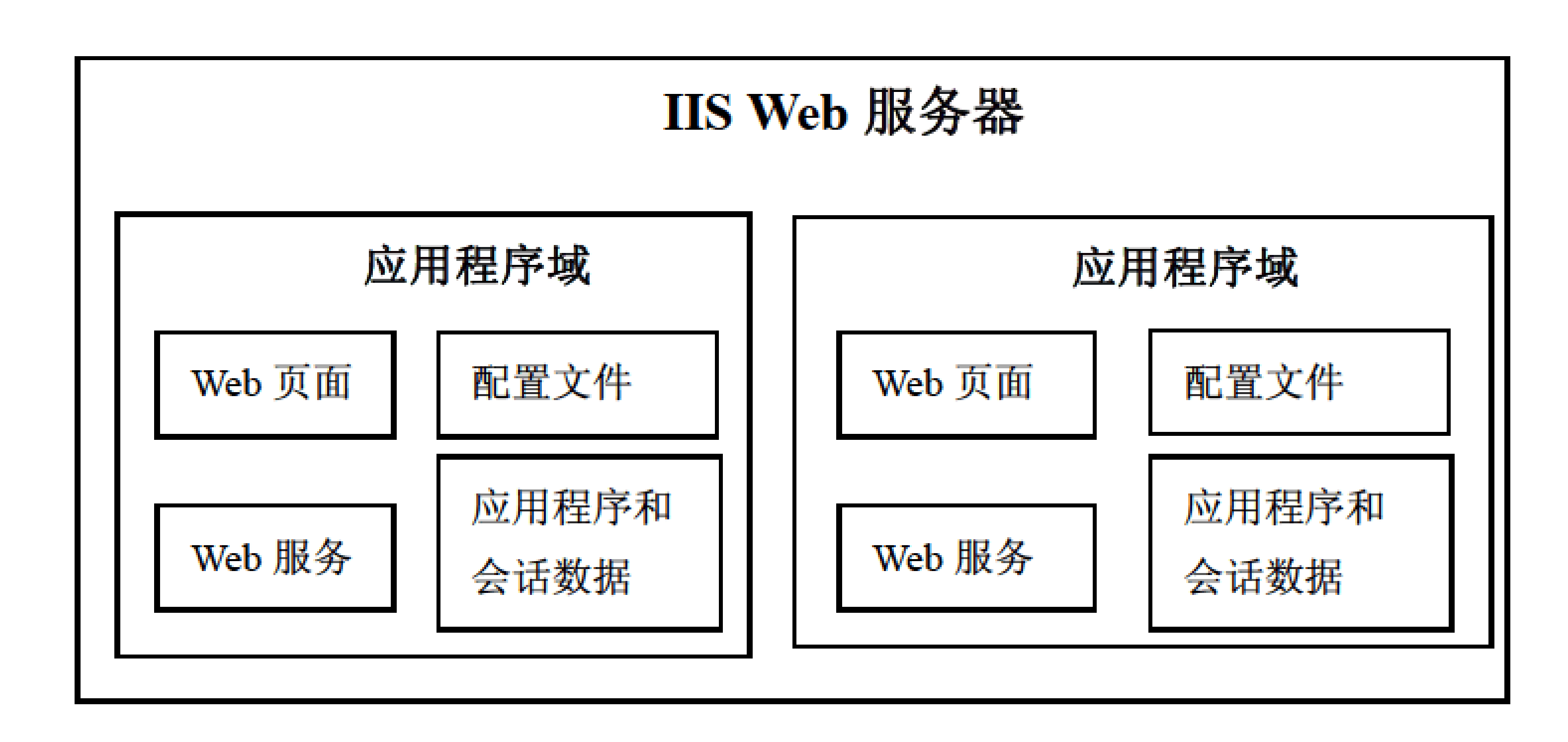


图 2-1　ASP.NET 应用程序域

2.1.1　ASP.NET 的文件类型

ASP.NET 应用程序可以包含很多种不同类型的文件，而不同类型的文件提供了不同的功能。在“添加新项”对话框中包含了允许向 Web 站点中添加的文件类型。这些能够添加到站点中的文件可以分组到不同的类别中。下面将讨论其中最重要的文件。

1. Web 文件

Web 文件是 Web 应用程序中特有的文件，可以由浏览器直接请求，也可以用来构建在浏览器中请求的 Web 页面的一部分。如表 2-1 所示为各种 Web 文件的扩展名及其描述。

表 2-1　ASP.NET Web 文件类型

文 件 类 型	描　　述
.aspx 文件	.aspx 文件是 ASP.NET 的 Web 页面文件，与传统 ASP 应用程序中的.asp 文件对应。.aspx 文件中包含页面的用户界面，也可以包含一些基本的应用程序代码。用户向 Web 服务器请求某个.aspx 文件，或者导航到某个.aspx 文件时，将进入一个 Web 应用程序
.ascx 文件	.ascx 文件是 ASP.NET 的用户控件。用户控件与 Web 页面类似，但用户无法直接访问.ascx 文件。实际上，.ascx 文件必须嵌入到一个 ASP.NET 页面中才能运行。可以使用用户控件来开发一些用户界面的模块，并在其他页面中直接使用这些用户控件，而不必重复编写相同的代码
.master 文件	.master 文件是母版页文件，这些文件允许定义 Web 站点的全局结构和外观
.htm/.html 文件	可用来显示 Web 站点中的静态 HTML
.css 文件	层叠样式表文件，包含允许定制 Web 站点的样式和格式的 CSS 代码
web.config 文件	web.config 文件是一个基于 XML 的 ASP.NET 配置文件。该配置文件包含了自定义的安全设置、状态管理和内存管理等多种配置的设置。本书中很多地方都会用得该文件的配置
.sitemap 文件	站点导航文件，包含一个层次结构，表示站点中 XML 格式的文件，用于导航控件

(续表)

文 件 类 型	描　　述
.js 文件	Jscript 文件，包含可以在客户端浏览器中执行的 JavaScript(Microsoft 称之为 JScript)
.skin 文件	外观文件，包含 Web 站点中的控件设计信息

2. 代码文件

ASP.NET 的代码文件如表 2-2 所示。

表 2-2　ASP.NET 代码文件类型

文 件 类 型	描　　述
.asmx 文件	.asmx 文件是 ASP.NET Web 服务文件。Web 服务是一组方法的集合，这些方法可以通过 Internet 进行调用。其工作原理与 Web 页面完全不同，但是 Web 服务与 Web 页面共享同一应用程序中的各种资源、配置设置和内存区域
Global.asax 文件	Global.asax 文件是一个全局应用程序文件。在 Global.asax 文件中，可以定义全局变量(全局变量可以在当前 Web 应用程序的所有页面中访问)，也可以定义应用程序的全局事件(如当一个 Web 应用程序启动时的事件)
.cs 文件	.cs 文件是 Web 页面的后置代码文件，其中包含执行页面逻辑功能的 C#代码。.cs 文件将应用程序的逻辑从 Web 页面的用户界面中分离出来

注意：

如果使用的是 Visual Basic.NET 语言，则对应的后置代码文件扩展名是.vb。

3. 数据文件

数据文件用来存储可以用在站点和其他应用程序中的数据。这组文件由 XML 文件、数据库文件以及与使用数据相关的文件组成，如表 2-3 所示。

表 2-3　ASP.NET 数据文件类型

文 件 类 型	描　　述
.xml 文件	.xml 文件用来存储 XML 格式的数据。除了纯 XML 文件外，ASP.NET 还支持几种基于 XML 的文件，其中包括前面提到的 web.config 和 Site Map 文件
.mdf 文件	.mdf 文件是 Microsoft SQL Server 使用的数据库文件
.dbml 文件	.dbml 文件用于声明性地访问数据库，不需要写代码。从技术上来讲，这并不是一个数据文件，因为它不包含实际数据。然而，由于它们与数据库绑定得如此紧密，因此把它们归组在这个标题下是有意义的

2.1.2　ASP.NET 应用程序的目录结构

每个 Web 应用程序都应该具有一个良好的目录结构规划。开发者在对程序进行设计时应该将特定类型的文件存放在某些文件夹中，以方便在今后开发中的管理和操作。ASP.NET 保留了一些特殊的子目录，程序开发人员可以直接使用，并且还可以在应用程序中增加任意多个文件和文件夹。

1. App_Code 子目录

App_Code 子目录在 Web 应用程序根目录下，它存储所有应当作为应用程序的一部分动态编译的类文件。这些类文件自动链接到应用程序，而不需要在页面中添加任何显式指令或声明来创建依赖性。App_Code 目录中放置的类文件可以包含任何可识别的 ASP.NET 组件，包括自定义控件、辅助类、build 提供程序、业务类、自定义提供程序和 HTTP 处理程序等。

在开发时，对 App_Code 目录的更改会导致整个应用程序的重新编译。对于大型项目，这可能不受欢迎，而且很耗时。为此，鼓励大家将代码进行模块化处理到不同的类库中，按逻辑上相关的类集合进行组织。应用程序专用的辅助类大多放置在 App_Code 文件夹中。

注意：

App_Code 目录中存放的所有类文件应当使用相同的语言。如果类文件使用两种或多种语言编写，则必须创建特定语言的子目录，以包含用各种语言编写的类。一旦根据语言组织这些类文件，就要在 web.config 文件中为每个子目录添加设置，有关 web.config 文件的使用将在后面进行详细介绍。

2. App_Data 子目录

App_Data 子目录保存应用程序使用的数据库。它是一个集中存储应用程序所用数据库的地方。该目录是 ASP.NET 为程序提供存储自身数据的默认位置，但该文件夹内容不由 ASP.NET 处理。它通常以文件(诸如 Microsoft Access 或 Microsoft SQL Server 数据库、XML 文件、文本文件以及应用程序支持的任何其他文件)的形式对数据进行存储。

当然，也可以将数据文件保存到其他目录中。

提示：

默认情况下，ASP.NET 账户被授予对该子目录的完全访问权限。如果要改变 ASP.NET 账户，一定要确保新账户被授予对该目录的读、写访问权。

3. Bin 子目录

Bin 子目录包含应用程序所需的用于控件、组件或需要引用的任何其他代码的可部署程序集。该目录中存在的任何.dll 文件将自动链接到应用程序。可以在 Bin 目录中存储编译的程序集，并且在 Web 应用程序任意处的其他代码会自动引用该目录。例如：如果为自

定义类编译好了代码，那么就可以将编译后的程序集复制到 Web 应用程序的 Bin 目录中，这样，所有页就都可以使用这个类了。

Bin 目录中的程序集无须注册。只要.dll 文件位于 Bin 目录中，ASP.NET 就可以识别它。如果更改了.dll 文件，并将它的新版本写入到了 Bin 目录中，则 ASP.NET 也会检测到更新，并对随后的新页请求使用新版本的.dll 文件。

Bin 目录中程序集的作用范围为当前应用程序。因此，它们无法访问当前 Web 应用程序之外的资源或调用当前 Web 应用程序之外的代码。此外，在运行时，程序集的访问级别由本地计算机上指定的信任级别确定。

App_Code 子目录和 Bin 子目录是 ASP.NET 网站中的共享代码文件夹，如果 Web 应用程序要在多个页之间共享代码，就可以将代码保存在 Web 应用程序根目录下的这两个特殊目录中。当创建这些子目录并在其中存储特定类型的文件时，ASP.NET 将使用特殊方式处理。

4. App_GlobalResources 子目录

App_GlobalResources 子目录用于保存 Web 应用程序中的全局资源文件，资源文件是一些字符串表，当应用程序需要根据某些事情进行修改时，资源文件可用于这些应用程序的数据字典。可以在 App_GlobalResources 子目录中添加程序集资源文件(.resx)，它们会动态编译，成为解决方案的一部分，供程序中的所有.aspx 页面使用。在使用 ASP.NET 2.0/2.1 时，必须使用 resgen.exe 工具把资源文件编译为.dll 或.exe，才能在解决方案中使用。而从 ASP.NET 3.5 开始，资源文件的处理就容易多了。除了字符串之外，还可以在资源文件中添加图像和其他文件。

当需要开发一个支持多种语言的 Web 网站时，该目录用于进行本地化设置。

5. App_LocalResources 子目录

App_GlobalResources 子目录用于合并可以在应用程序范围内使用的资源。如果对构造应用程序范围内的资源不感兴趣，而对只能用于一个.aspx 页面的资源感兴趣，就可以使用该目录。可以把专用于页面的资源文件添加到该目录中，方法是构建.resx 文件名，如下所示：

```
Default.aspx.resx
Default.aspx.fi.resx
Default.aspx.ja.resx
Default.aspx.en-gb.resx
```

这样，就可以从 App_LocalResources 目录中的相应文件中检索在 Default.aspx 页面上使用的资源声明。如果没有找到匹配的资源，将默认使用 Default.aspx.resx 资源文件。

6. App_WebReferences 子目录

App_WebReferences 子目录用于保存当前 Web 应用程序中用到的 Web 服务引用。

7. App_Themes 子目录

App_Themes 子目录用于存放 Web 应用程序中使用的主题。主题是为站点上的每个页面提供统一外观和操作方式的一种新方法。通过 skin 文件、CSS 文件和站点上服务器控件使用的图像来实现主题功能。所有这些元素都可以构建一个主题，并存储在解决方案的 App_Themes 目录中。

当用户试图将一个文件添加到它的特色文件夹之外时，VWD 就会发出警告，并询问用户是否创建特色文件夹并将文件放在其中。如图 2-2 所示，是试图在 Web 站点的根目录中添加一个外观文件时弹出的提升对话框。

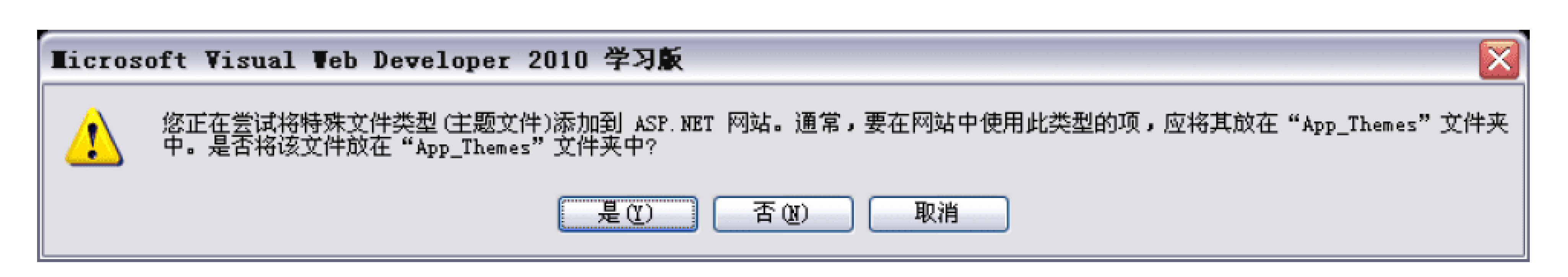


图 2-2　添加特殊文件时的提示对话框

2.2　ASP.NET 的内置对象

ASP.NET 能够成为一个庞大的软件体系，与它提供了大量的对象类库有很大的关系。这些类库中包含许多封装好的内置对象，开发人员可以直接使用这些对象的方法和属性。主要包括：Page、Response、Request、Application、Session、Server、ViewState 和 Cookie 等。下面将分别介绍这些对象的常用方法及属性。

2.2.1　Page 类与 Web 窗体页指令

在 ASP.NET Framework 中，Page 类为 ASP.NET 应用程序文件所构建的对象提供基本行为。该类在 System.Web.UI 命名空间中，从 TemplateControl 类派生而来，而 TemplateControl 类继承自 System.Web.UI.Control，它也是一种特殊的 Control 类，并实现了 IHttpHandler 接口。

1. Page 对象

Page 对象对应 Web 窗体，主要用来设置与网页有关的各种属性、方法和事件。Page 对象充当页面中所有服务器控件的命名容器。

在页面工作过程中，每个页面都被编译为一个类，当有请求的时候就会对这个类进行实例化。对于页面的生存周期，Page 对象一共要关心以下 5 个阶段。

- 页面初始化：在这个阶段，页面及其控件被初始化，页面确定这是一个新的请求还是一个回传请求。页面事件处理器 Page_PreInit 和 Page_Init 被调用。另外，所有服务器控件的 PreInit 和 Init 被调用。

- 载入：经过页面初始化之后，页面将进入载入阶段。在该阶段，如果当前页面的请求是一个回传请求，则该页面将从视图状态和控件状态中加载控件的属性。在此过程中，页面将引发 Load 事件。
- 回送事件处理：如果请求是一个回传请求，任何控件的回发事件处理过程都将被调用。
- 呈现：在页面呈现状态中，视图状态被保存到页面。页面和控件的 PreRender 和 Render 方法先后被调用。最后，呈现的结果通过 HTTP 响应发送回客户端。
- 卸载：对页面使用过的资源进行最后的清除处理，控件或页面的 Unload 方法将被调用。

说明：

单一页面在运行时被视为继承 Page 类。而在后台代码模式中，后台的.cs 文件中包含一个继承在 Page 类的分部类，即具有 partial 关键字的类声明，在对代码分离页进行编译时，ASP.NET 基于，aspx 文件生成一个分部类，该类是.cs 文件中定义的分部类的另一部分，两者一起编译成程序集，运行该程序集可以输出程序到浏览器。

Page 类的常用属性如表 2-4 所示。

表 2-4　Page 类的属性

属　性	描　述
Application	只读属性，为当前 Web 请求获取 HttpApplicationState 对象
Cache	只读属性，获取与网页所在的应用程序相关联的 cache 对象
EnableViewState	当前网页结束时，是否要保持视图状态及所包含的服务器的视图状态，默认为 True
IsPostBack	只读属性，指示该页是否正为响应客户端回发而加载，还是正被首次加载和访问
IsValid	只读属性，检查网页中的控件是否全部验证成功，全部验证成功返回 True，否则返回 False
Request	获取请求的页的 HttpRequest 对象
Response	获取与该 Page 对象关联的 HttpResponse 对象
Server	获取 Server 对象，它是 HttpServerUtility 类的实例
Session	获取 ASP.NET 提供的当前 Session 对象
Validators	获取请求的页上包含的全部验证控件的集合
ViewState	获取状态信息的字典，这些信息使用户可以在同一页的多个请求间保存和还原服务器控件的视图状态

Page 类的主要方法和事件如表 2-5 所示。

表 2-5　Page 类的主要方法和事件

方法或事件	描　述
DataBind 方法	将数据源与 Web 上的服务器控件进行绑定
Dispose 方法	让服务器控件在释放内存前执行清理工作
FindControl 方法	在 Web 窗体上搜索标识为指定 id 的服务器控件
HasControl 方法	如果 Page 对象中包含服务器控件，则返回 True，否则返回 False
MapPath 方法	将虚拟路径转换为实际路径
PreInit 事件	在页初始化开始时发生
PreLoad 事件	在页的 Load 事件之前发生
Load 事件	当服务器控件加载到 Page 对象中时发生
Init 事件	当服务器控件初始化时发生，初始化是控件生存期的第一步
PreRender 事件	在加载 Control 对象之后、呈现之前发生
InitComplete 事件	在页初始化完成时发生
LoadComplete 事件	在页生存周期的加载阶段结束时发生
UnLoad 事件	当服务器控件从内存中卸载时发生
DataBinding 事件	当数据源与页面上的服务器控件进行绑定时发生

Page 对象的事件贯穿页面执行的整个过程。大多数情况下，只需关心 Page_Load 事件即可。由于 Page_Load 方法会在每次页面被加载时执行，所以，即使是回传的情况下也会调用该方法，此时，可以使用 Page 对象的 IsPostBack 属性来判断是否是回传请求，从而进行不同的处理。例 2-1 演示了 Page 类各事件发生的时刻。

例 2-1：演示加载页面时，Page 类各事件的发生顺序。

(1) 启动 VWD2010，选择“文件”|“新建网站”命令，新建一个空网站 Chapter2，单击“确定”按钮。

(2) 选择“网站”|“添加新项”命令，打开“添加新项”对话框，添加一个名为 Default.aspx 的 Web 窗体。

(3) 切换到页面的“设计”视图，从“工具箱”中拖动一个 Button 控件和一个 Label 控件到 Web 窗体中，系统自动为它们命名为 Button1 和 Label1。在 Button 控件的上方添加文本信息“Page 事件演示”，效果如图 2-3 所示。

技巧：

在“工具箱”中双击相应的控件，即可将其添加到 Web 窗体中。

(4) 选中 Label 控件，在“属性”窗口中修改其 Text 属性为“下面将显示各事件发生的顺序”。

(5) 在 Web 窗体中双击 Button 控件，即可为其添加单击事件处理程序，同时将跳转到代码文件 Default.aspx.cs 的相应位置，并在此添加如下代码，修改 Label 控件的 Text 属性。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label2.Text += "<br>按钮的单击事件处理程序";
}
```

技巧：

也可以选中 Button 控件，然后在“属性”窗口中单击工具栏中的“事件”按钮，然后从事件列表中找到 Click 事件来添加其事件处理程序，如图 2-4 所示。

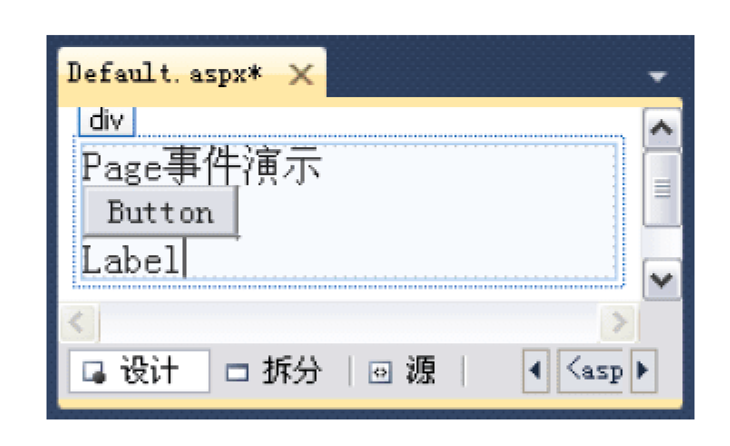


图 2-3　添加控件到 Web 窗体

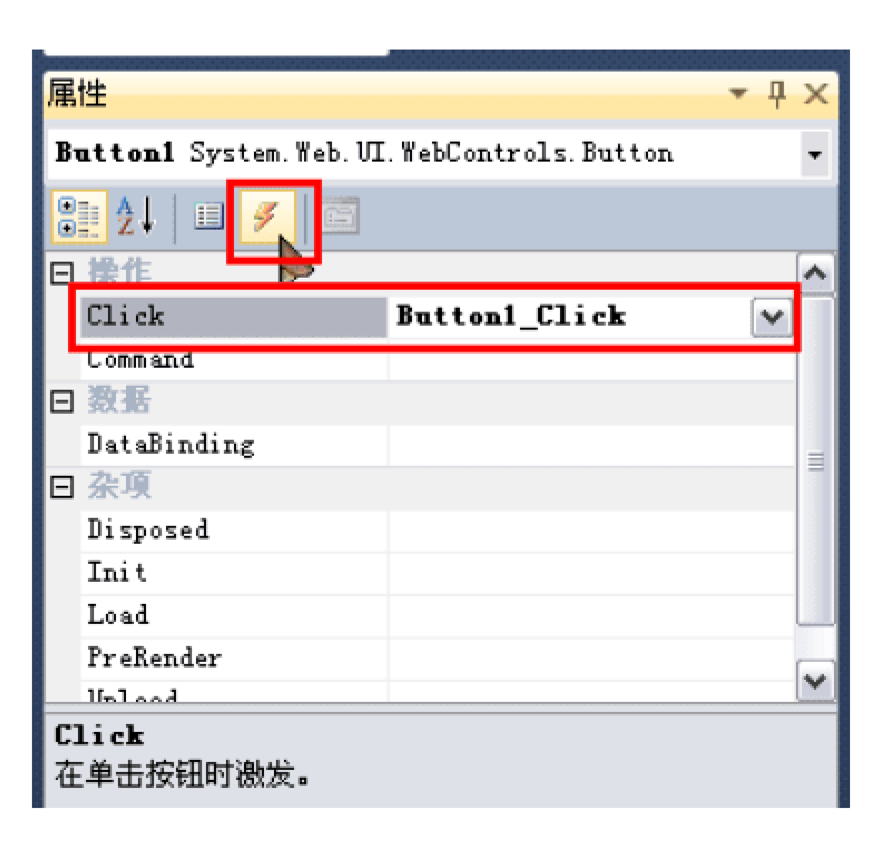


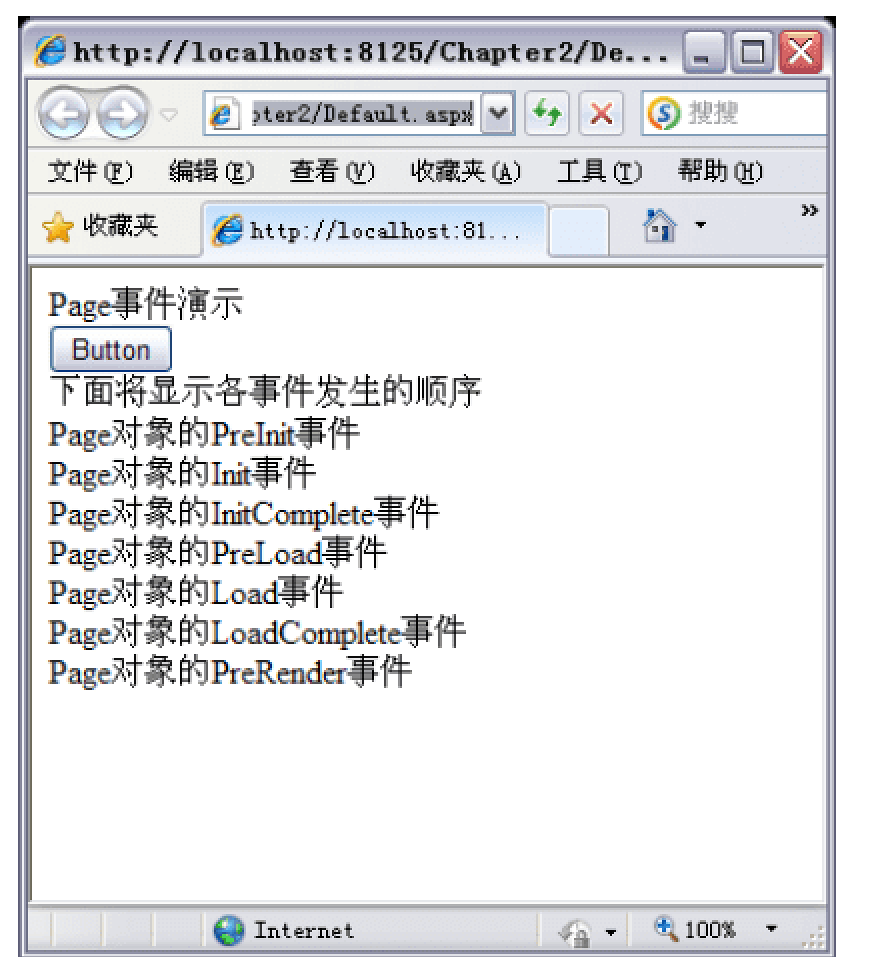
图 2-4　为控件添加事件处理程序

(6) 在代码中同时添加其他的 Page 对象的事件处理程序，如下所示：

```
protected void Page_InitComplete(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 InitComplete 事件";
}
protected void Page_PreInit(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 PreInit 事件";
}
protected void Page_Init(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 Init 事件";
}
protected void Page_PreLoad(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 PreLoad 事件";
}
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 Load 事件";
}
protected void Page_PreRender(object sender, EventArgs e)
```

```
        {
            Label1.Text += "<br>Page 对象的 PreRender 事件";
        }
        protected void Page_LoadComplete(object sender, EventArgs e)
        {
            Label1.Text += "<br>Page 对象的 LoadComplete 事件";
        }
```

(7) 编译并运行程序，或者按 F5 键，在 IE 中显示的运行效果如图 2-5 所示。



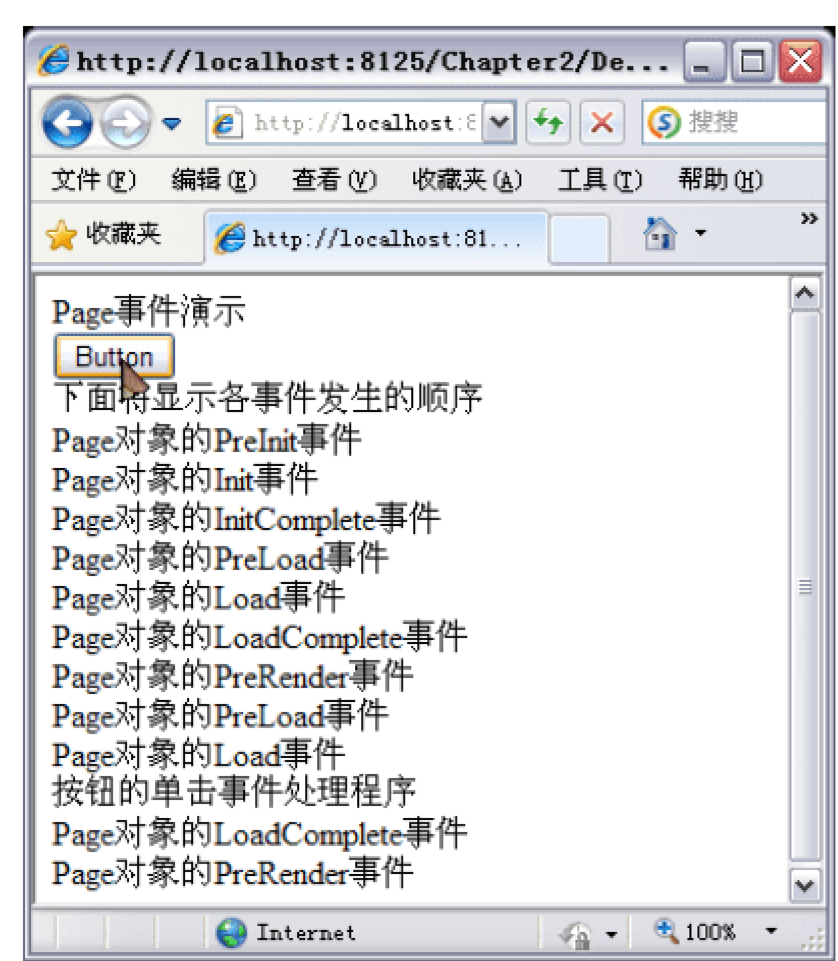


图 2-5　Page 对象演示结果

2. Web 窗体页指令

Web 窗体页指令也称为预编译指令，用来指定当请求 ASP.NET 页和用户控件时使用的设置。常用的预编译指令如表 2-6 所示。

表 2-6　常用的窗体页指令

指　　令	描　　述
@Page	该指令定义 ASP.NET 页分析器和编译器使用的页的特定属性，它只能包含在.aspx 文件中，且每个.aspx 文件中只能有一个@Page 指令
@Control	该指令定义 ASP.NET 页分析器和编译器使用的用户控件(.ascx 文件)特定的属性，它只能包含在.ascx 文件中，且每个.ascx 文件中只能有一个@Control 指令
@Assembly	该指令在编译过程中将程序集链接到当前页，以使程序集的所有类和接口都可用在该页上
@Implements	该指令指示当前页或用户控件实现指定的.NET 框架接口
@Import	该指令将命名空间显示导入到页中，使导入的命名空间的所有类和接口可用于该页，导入的命名空间可以是.NET 框架类库或用户定义的命名空间的一部分

(续表)

指　　令	描　　述
@OutputCache	该指令以声明的方式控制 ASP.NET 页或页中包含的用户控件的输出缓存策略
@Reference	该指令以声明的方式指示另一个用户控件或页源文件应该被动态编译并链接到在其中声明该指令的页
@Register	该指令将别名与命名空间及类名关联起来，以便在自定义控件语法中使用简明的表示法

2.2.2　Request 对象

Request 对象是 ASP.NET 当中最有用的对象之一，它与 Response 对象一起使用，达到沟通客户端与服务器端的作用，使它们之间可以很简单地交换数据。

Request 对象接收客户端通过表单或 URL 地址串发送来的变量，同时，也可以接收其他客户端的环境变量，如浏览器的基本情况、客户端的 IP 地址等。所有从前端浏览器通过 HTTP 通信协议送往后端 Web 服务器的数据，都是借助 Request 对象完成的。

Request 对象是 System.Web.HttpRequest 类的实例，当用户请求页面时，ASP.NET 将自动创建 Request 对象。

1. Request 对象的属性

Request 对象的常用属性如表 2-7 所示。

表 2-7　Request 对象的常用属性

属　　性	描　　述
ApplicationPath	获取 ASP.NET 应用程序虚拟目录的根目录
Browser	获取和设置客户端浏览器的兼容性信息
ContentEncoding	获取 Request 对象的编码方式
ContentLength	获取客户端发送信息的字节数
ContentType	获取和设置请求的 MIME 类型
Cookies	获取客户端 Cookie
FilePath	当前请求的虚拟路径
Files	获取客户端上传的文件集合
Form	获取表单变量集合
Headers	获取 HTTP 头信息
HttpMethod	HTTP 数据传输方法，如 GET、POST
Path	获取当前请求的虚拟路径
PhysicalPath	获取请求的 URL 物理路径
QueryString	获取查询字符串集合

(续表)

属　性	描　述
ServerVariables	获取服务器变量集合
TotalBytes	获取输入文件流的总大小
Url	获取当前请求的 URL
UrlReferrer	获取该请求的上一次请求的页面
UserAgent	客户端浏览器信息
UserHostAddress	客户端 IP 地址
UserHostName	客户端 DNS 名称
UserLanguages	客户端语言

2. 使用 Form 和 QueryString 集合传递数据

ASP.NET 是使用表单(Form)来实现用户数据提交的。对于 HTML 表单，可以使用 Get 方法或 Post 方法来实现数据提交。如果使用 Get 方法，就要使用 Request 对象的 QueryString 集合来得到相关的信息；如果使用 Post 方法，就要使用 Request 对象的 Form 集合来得到相关信息。下面分别讲解如何使用 Get 方法和 Post 方法。

- Get 方法：使用 Get 方法进行数据提取时，用户要提交的信息往往是作为查询字符串加在 URL 的后面传给接收程序的，一般限制在 2KB 左右。例如：http://www.domain.com/test.aspx?name=myname&password=mypassword。这样，在服务器端就可以使用 QueryString 集合对象来获取数据：

```
Request.QueryString["name"]
Request.QueryString["password"]
```

- Post 方法：使用 Post 方法时，用户浏览器的地址栏中不会显示相关的查询字符串。因此，如果需要提交的数据很多时，应使用 Post 方法，因为它对数据的大小和长度没有什么限制。另外，由于地址栏中不显示相关的查询字符串，所以使用 Post 方法就十分适合用来传递保密信息，例如用户的账号和密码。使用 Post 方法提交的数据，在服务器端可以使用 Form 集合对象来获取，其语法格式如下所示：

```
Request.Form
Request.Form["name"]
Request.Form.Get(Index)
```

3. Request 对象的方法

Request 对象的常用方法如表 2-8 所示。

表 2-8　Request 对象的常用方法

方　法	描　述
BinaryRead	以二进制方式读取指定字节的输入流
MapPath	为当前请求将请求的 URL 中的虚拟路径映射到物理路径
SaveAs	保存 HTTP 请求到硬盘
ValidateInput	验证客户端输入的数据，如果具有潜在的风险，则将会引发一个异常

2.2.3　Response 对象

Response 对象用于向客户端浏览器发生数据，告诉浏览器回应内容的报头、服务器端的状态信息以及输出指定的内容。用户可以使用该对象将浏览器的数据以 HTML 的格式发送到用户端的浏览器，它与 Request 组成一对接收、发送数据的对象，这也是实现动态的基础。

Response 对象实际是 System.Web 命名空间中的 HttpResponse 类的实例。CLR 会根据用户的请求信息建立一个 Response 对象。

1. Response 对象的属性

Response 的常用属性如表 2-9 所示。

表 2-9　Response 对象的常用属性

属　性	描　述
Buffer	获取或设置是否缓冲输出。如有缓冲，服务器在所有当前处理的页面的语句被处理之前不将 Response 送往客户端，除非有 Flush 或 End 方法被调用。True 表示需要，False 表示不需要，默认值是 True
Cache	获取缓存信息
CharSet	获取和设置输出流的 HTTP 字符集
ContentType	获取和设置输出流的 MIME 类型，默认值为：text/html，常用的类型还有：text/xml、text/plain、image/gif、image/jpeg
Cookie	获取 Cookie 集合
Expires	获取和设置浏览器缓存超时时间
IsClientConnected	获取客户端是否和服务器连接
Status	设置返回给客户端的状态
StatusCode	获取和设置返回给客户端状态字符串
StatusDescription	获取和设置状态说明

2. Response 对象的方法

Response 对象的常用方法如表 2-10 所示。

表 2-10　Response 对象的常用方法

方　法	描　述
AddHeader	添加 HTTP 头信息
AppendCookie	添加一个 Cookie
AppendHeader	追加 HTTP 头信息
AppendToLog	添加自定义信息到 IIS 日志中
BinaryWrite	以二进制的方式输出
Clear	清除输出缓存
Close	关闭和客户端的 Socket 连接
End	发送所有缓冲到客户端，并且停止执行页面
Flush	发送所有缓存到客户端
Redirect	重新定向 URL
RedirectPermanent	执行从所请求 URL 到所指定 URL 的永久重定向，并提供用于完成响应的选项
SetCookie	更新一个已有的 Cookie
Write	输出信息
WriterFile	直接将指定文件写到输出流

3. 重定向到另一个页面

以编程的方式将用户重定向到另一个页面，在 ASP.NET 中是非常有用和普遍的。例如，在一个第三方支付页面，一旦支付成功了，将在几秒钟之后自动跳转到商家页面。

ASP.NET 支持 3 种通过编写程序将用户重定向到新页面的主要方式：其中两种就是使用 Response 对象的 Redirect 和 RedirectPermanent(ASP.NET 4 中新增的选项)方法，而第 3 种则是使用 2.2.5 节将要介绍的 Server 对象的 Transfer 方法，它是在客户端执行的。此处重点介绍前两种方法。

在每个 ASPX 页面内，用户都具有对 Response 属性的访问权限，Response 对象的 Redirect 和 RedirectPermanent 方法会向浏览器发送请求新页面的指令。每个 Redirect 方法能以两种不同的方式使用：

```
Response.Redirect(newUrl)
Response.Redirect(newUrl, endResponse)
Response.RedirectPermanent(newUrl)
Response.RedirectPermanent(newUrl, endResponse)
```

Redirect 和 RedirectPermanent 的区别主要与搜索引擎优化有关。使用 Redirect 是告诉客户端，页面只是被临时移动了。通常使用该方法基于某些动作将用户重定向到一个新页面上。例如，在填写完注册信息后，可能需要将用户导航到一个注册成功页面或是注册并登录成功后的页面。

RedirectPermanent 用于告诉客户端，页面被永久移动了。例如，网站中有一个不再使用的页面 Index.aspx。搜索引擎可能会一直请求这个页面。如果将下面的代码添加到 Index.aspx 后台代码文件中，客户端(包括搜索引擎)就会被导航到 Default.aspx 页面。此时，搜索引擎会记录永久性重定向的节点，并会停止对 Index.aspx 页面的请求，且定位到 Default.aspx 页面上。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.RedirectPermanent("Default.aspx");
}
```

说明：

重定向方法的另一个版本有一个名为 endResponse 的附加 Boolean 参数，当 endResponse 参数为 False 时，它允许在重定向动作完成后执行余下的所有代码。这并不是必需的，但在默认情况下使用它来结束响应。

4. Request 和 Response 对象配合使用

下面的例子演示了 Request 和 Response 对象的配合使用。

例 2-2：利用 Request 对象和 Response 对象进行数据传送，理解 GET 和 POST 方法的不同，以及数据的获取方式。

(1) 启动 VWD 2010，打开网站 Chapter2，通过“添加新项”对话框在该网站中添加一个名为 Request.aspx 的 Web 窗体页。

(2) 在 Request.aspx 页面的<body>标签中添加如下代码：

```
<body>
    <h2>登录页面</h2>
    <form id="form1" action="Response.aspx" method="post">
    姓名：<input name="user" type="text" />
    <br>密码：<input name="password" type="text" />
    <br><input name="submit" type="submit" value="POST 方式提交" />
    </form>
    <form id="form2" action="Response.aspx" method="get">
    <br>姓名：<input name="user" type="text" />
    <br>密码：<input name="password" type="text" />
    <br><input name="submit" type="submit" value="GET 方式提交" />
    </form>
</body>
```

在 Request.aspx 页面中，定义两个表单：form1 和 form2，分别使用 POST 和 GET 方式提交数据到 Response.aspx 页面，下面我们就来创建 Response.aspx 页面。

(3) 在“解决方案资源管理器”窗口中，右击解决方案，从弹出的快捷菜单中选择“添加新项”命令，添加名为 Response.aspx 的 Web 窗体。

(4) 在 Response.aspx 页面中的<body>标签中添加如下代码：

```
<body>
    <form id="form1" runat="server">
    <div>
    <asp:button runat="server" text="返回登录页面" onclick="Button1_Click" />
    </div>
    </form>
</body>
```

(5) 在 Response.aspx.cs 代码文件中，添加页面的 Load 事件和按钮的单击事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.HttpMethod.Equals("GET"))
    {
        Response.Write("以下信息来自于 Request 页面,数据传输方法为 GET<br>");
        Response.Write("QueryString：" + Request.QueryString);
        Response.Write("<br>用户名:" + Request.QueryString["user"]);
        Response.Write("<br>密码:" + Request.QueryString["password"]);
    }
    if (Request.HttpMethod.Equals("POST"))
    {
        Response.Write("以下信息来自于 Request 页面,数据传输方法为 POST<br>");
        Response.Write("<br>用户名:" + Request.Form["user"]);
        Response.Write("<br>密码:" + Request.Form["password"]);
    }
}
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Redirect("Request.aspx");
}
```

现在，站点中一共有 3 个 Web 页面了，那么，如何确定站点的起始页呢？下面介绍如何设置站点的启动选项。

(6) 选择“视图”|“属性页”命令，或者在“解决方案资源管理器”窗口中右击解决方案，从弹出的快捷菜单中选择“属性页”命令，打开网站的“属性页”对话框，选择“启动选项”，如图 2-6 所示，默认的启动选项为当前页。

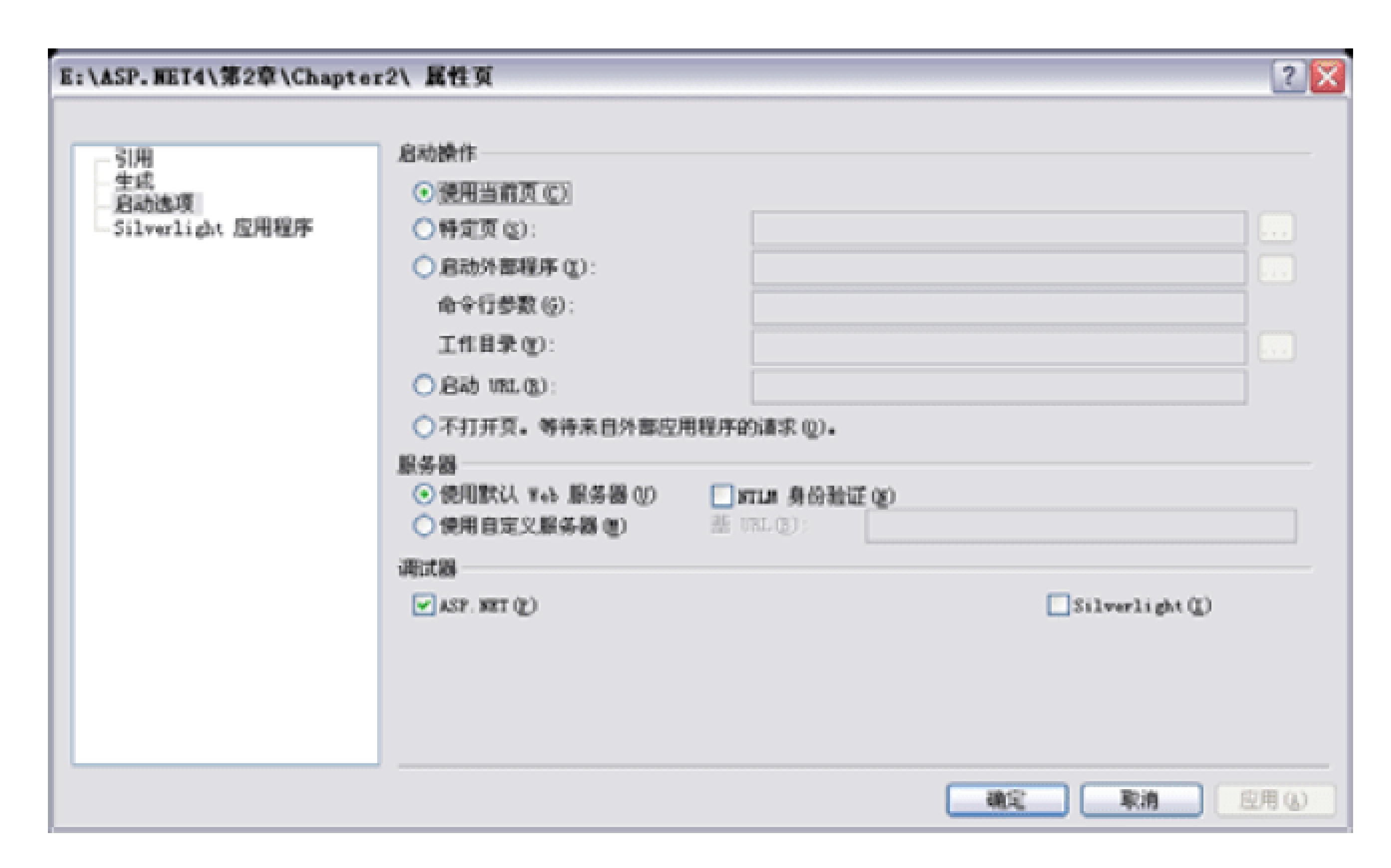


图 2-6　网站的“属性页”对话框

(7) 在本例中，设置启动页为 Request.aspx，在右侧区域中选中“特定页”单选按钮，此时，后面的文本框和浏览按钮会被激活，可直接在文本框中输入 Request.aspx；也可以单击按钮，从弹出的对话框中选择 Request.aspx 页面，如图 2-7 所示。

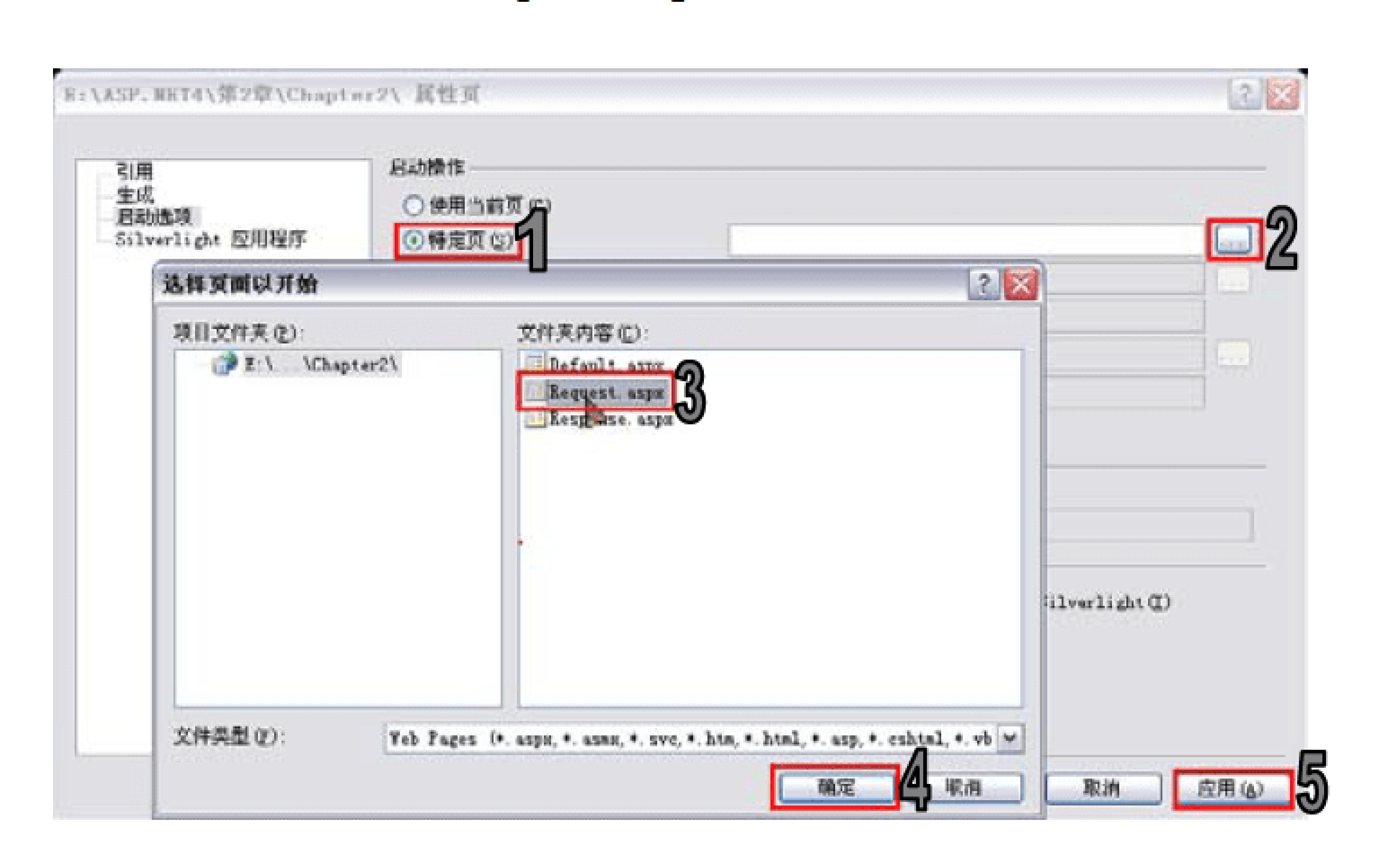


图 2-7　设置启动页为 Request.aspx 页面

(8) 编译并运行程序，在浏览器中加载 Request.aspx 页，如图 2-8 所示。

(9) 在上方的两个文本框中分别输入“姓名”和“密码”，然后单击“POST 方式提交”按钮，跳转到 Response.aspx 页面，如图 2-9 所示。

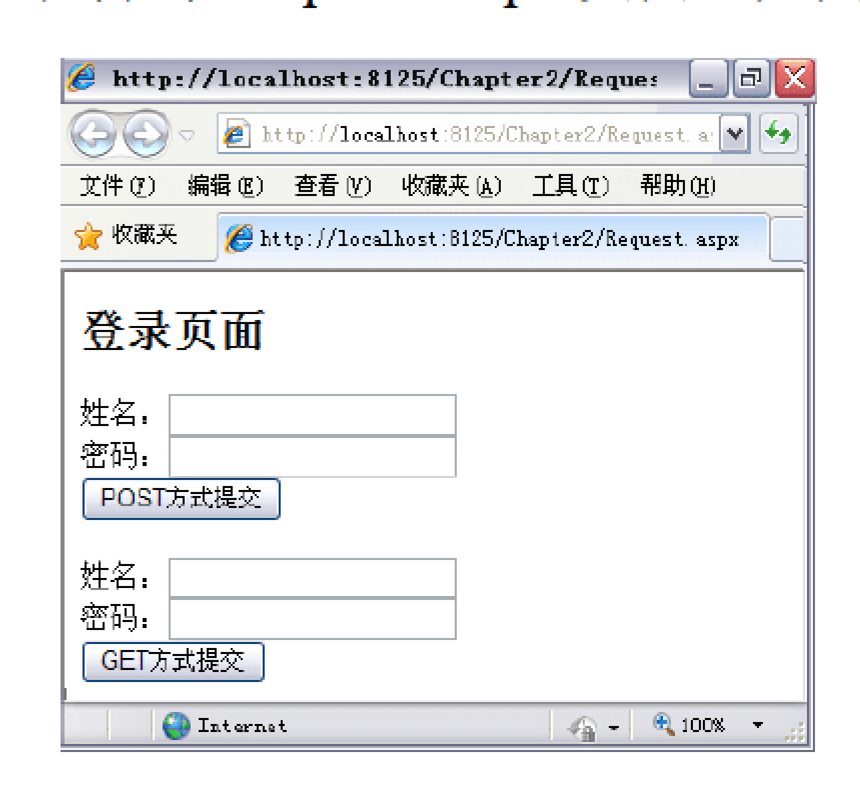


图 2-8　Request.aspx 页面

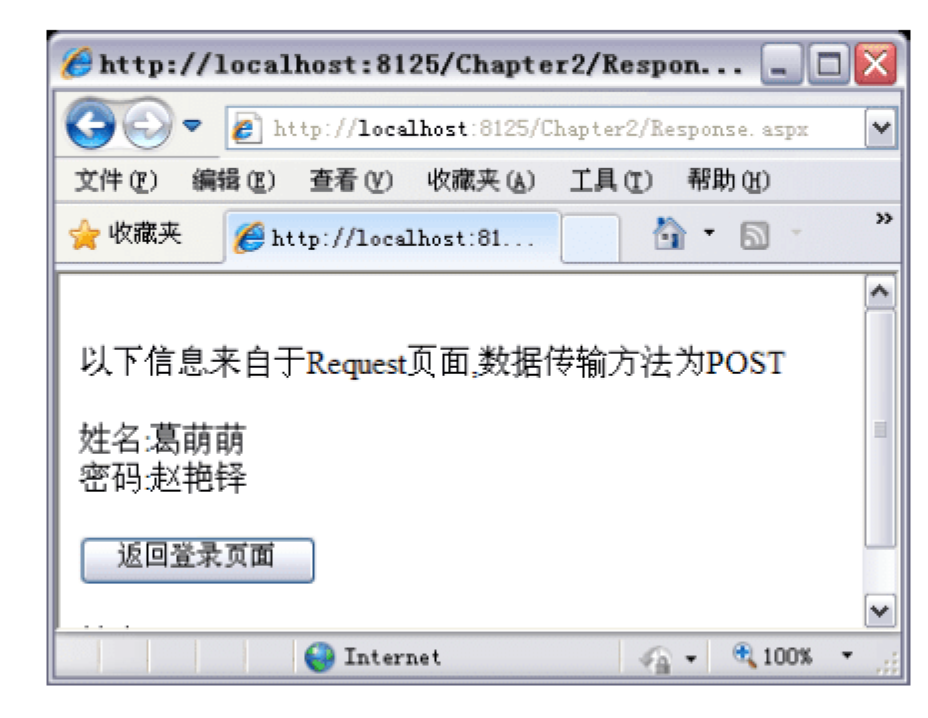


图 2-9　以 POST 方式传送数据

(10) 单击“返回登录页面”按钮，返回到 Request.aspx 页面，在下面的文本框中输入“姓名”和“密码”，单击“以 GET 方式提交”按钮，此时得到的页面如图 2-10 所示，可以在“标题栏”和“地址栏”中看到传递的参数。

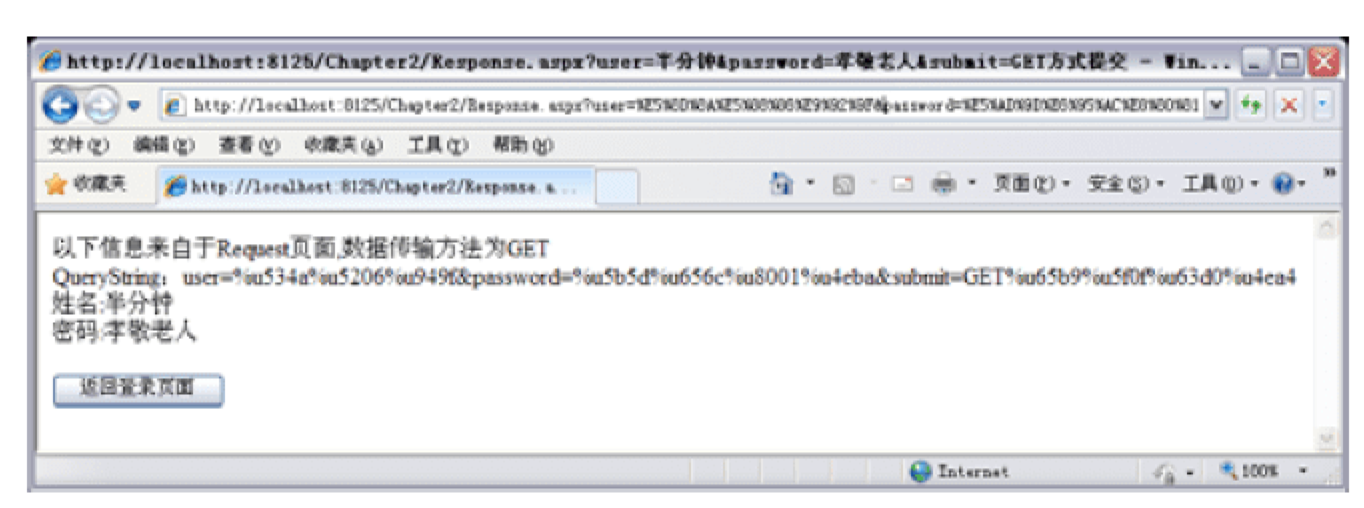


图 2-10　以 GET 方式传送数据

2.2.4　Application 对象

Application 对象用来保存希望在多个页面之间传递的变量。由于在整个应用程序生存周期中，Application 对象都是有效的，所以在不同的页面中都可以对它进行存取，就像使用全局变量一样方便。

在 ASP.NET 环境中，Application 对象是 System.Web.HttpApplicationState 类的实例，它可以在多个请求、链接之间共享公用信息，也可以在各个请求链接之间充当信息传递的管道。

Application 对象可以建立 Application 变量，它和一般程序变量不同，Application 变量是一个 Contents 集合对象，此变量可以为访问网站的每位用户提供一个共享数据的通道，因为 Application 变量允许网站的每位用户获取或更改其值。

说明：

Application 对象在第 1 个 Session 对象(将在 2.2.6 节介绍)建立后创建，直到 Web 服务器关机后所有的用户都离线后才会删除。

1. Application 对象的属性

Application 对象的常用属性如表 2-11 所示。

表 2-11　Application 对象的常用属性

属　　性	描　　述
AllKeys	获得访问 HttpApplicationState 集合的所有键
Contents	获得 HttpApplicationState 对象的引用
Count	获得 HttpApplicationState 集合的数量
Item	通过名称和索引访问 HttpApplicationState 集合
Keys	获得访问 HttpApplicationState 集合的所有键，从 NameObjectCollectionBase 继承
StaticObjects	获得所有使用<object>标签声明的应用程序集对象

2. Application 对象的方法

Application 对象的常用方法如表 2-12 所示。

表 2-12　Application 对象的常用方法

方　法	描　述
Add	添加一个新的对象到 HttpApplicationState 集合
Clear	清除 HttpApplicationState 集合中的所有对象
Get	通过索引和名字获得 HttpApplicationState 对象
GetKey	通过索引获得一个 HttpApplicationState 名称
Lock	锁定访问 HttpApplicationState 变量
UnLock	取消锁定，一般情况下需要操作 Application 变量则设置为 Lock，操作完成后则设置为 Unlock
Remove	从 HttpApplicationState 集合删除一个对象
RemoveAll	删除 HttpApplicationState 集合所有对象
RemoveAt	根据索引删除一个 HttpApplicationState 对象
Set	更新一个 HttpApplicationState 变量

3. 使用 Application 对象

不论网站中有多少位用户同时浏览网站，在服务器端都只保留一份 Application 变量。可以使用如下语法格式来设置或读取 Application 变量：

```
Application["title"]="金百合拉丁舞培训学校"
Application.Set("count",(int)Application["count"] + 1);
```

ASP.NET 应用程序的每位用户都可以存取 Application 变量，用户可以同时读取 Application 变量，但是在同一时间只能允许一位用户修改 Application 变量，这就需要使用 Lock 和 Unlock 方法。在修改 Application 变量之前先使用 Lock 方法进行锁定，修改完以后再使用 Unlock 方法解锁。

例 2-3 将演示 Application 对象的使用。

例 2-3：使用 Application 对象统计网站访问人数。

(1) 启动 VWD 2010，选择“文件”|“打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 Application.aspx 的 Web 窗体，在“设计”视图中添加一个 Label 控件到 Web 窗体中，系统自动将它命名为 Label1。

(3) 在 Application.aspx.cs 文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Application.Lock();
    Application["count"] = (Convert.ToInt32(Application["count"]) + 1).ToString();
```

```
            Application.UnLock();
            Label1.Text = "您是第  " + Application["count"].ToString() + "  位访客";
        }
```

(4) 编译并运行程序，在浏览器中打开 Application.aspx 页面，如图 2-11 所示，刷新页面可以发现数字会增加。



图 2-11　Application 对象示例

注意：

在例 2-2 中，设置了网站的启动选项为 Request.aspx 页面，因此本例中启动网站默认打开的是 Request.aspx 页面，读者需要手动在地址栏中输入 Application.aspx 页面或修改网站的启动选项才能访问 Application.aspx 页面。

2.2.5　Server 对象

Server 对象即服务器对象，就是在服务器上工作的一个对象，它包含一些与服务器相关的信息，是 System.Web.HttpServerUtility 类的实例。使用它可以获取最新的有关出错的信息，在页面之间传递控件，对 HTML 进行编码和解码等。

1. Server 对象的属性

Server 对象的属性可以获取 Web 服务器名称和设置或获取超时时间，如表 2-13 所示。

表 2-13　Server 对象的常用属性

属　性	描　述
MachineName	获得服务器计算机名称
ScriptTimeout	获得和设置请求超时的事件

2. Server 对象的方法

Server 对象的方法可以获取文件路径、使用 COM 组件以及执行 HTML 和 URL 编码。其常用方法如表 2-14 所示。

表 2-14　Server 对象的常用方法

方　法	描　述
ClearError	清除前一个异常
CreateObject	建立一个 COM 组件对象的实例
Execute	执行指定资源并返回
GetLastError	获取最近一次异常
Transfer	结束当前页执行，转到其他页执行
HtmlEncode	将指定的字符串进行 HTML 编码
HtmlDecode	进行 HTML 解码
MapPath	对虚拟目录进行物理映射
UrlEncode	进行 URL 编码，以便通过 URL 从 Web 服务器到客户端进行可靠的 HTTP 传输
UrlDecode	进行 URL 解码
UrlPathEncode	对 URL 字符串的路径部分进行 URL 编码，并返回已编码的字符串

3. 服务器端重定向

如果要对用户的请求发出不同的页面，而又不想修改客户端的地址栏，那么，使用服务器端重定向就非常方便。这样可以隐藏页面名称和查询字符串的细节，从而产生从用户的角度来看比较干净的URL。这通常用在所谓的URL重写的情况中，用于创建干净的URL。例如，用户可能请求一个这样的页面：

```
http://www.yanzhao.com/Pictures/2012/03/
```

服务器在后台可能将其转换为：

```
http://www. yanzhao.com/Pictures/Display.aspx?year=2012&month=03
```

显然，第一个 URL 更容易理解和在浏览器中输入。除了容易理解外，服务器端重定向也可能稍稍加快站点的速度。不用向浏览器发送响应让它去请求一个新页面，而是直接将用户转到新页面上，以节省一些网络开销。

服务器端重定向使用的就是 Server 对象的 Transfer 方法。Server.Transfer 不是指示浏览器去获取一个新页面，而是完全在服务器上发生。放弃旧页面的输出，并启动新的页面生命周期。然后将新的页面生成的内容发送回浏览器，而保持浏览器的地址栏不变。下面举例说明 Response.Redirect 和 Server.Transfer 方法之间的区别。

例 2-4：使用服务器端重定向将用户导航到 Response.aspx 页面。

(1) 启动 VWD 2010，选择“文件”|“打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 Transfer2Response.aspx 的 Web 窗体。

(3) 无须设计页面，直接打开 Transfer2Response.aspx 页面的后台代码文件，在页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Server.Transfer("Response.aspx?user=匿名&password=密码");
}
```

(4) 编译并运行程序，在浏览器中打开 Transfer2Response.aspx 页面，效果如图 2-12 所示。

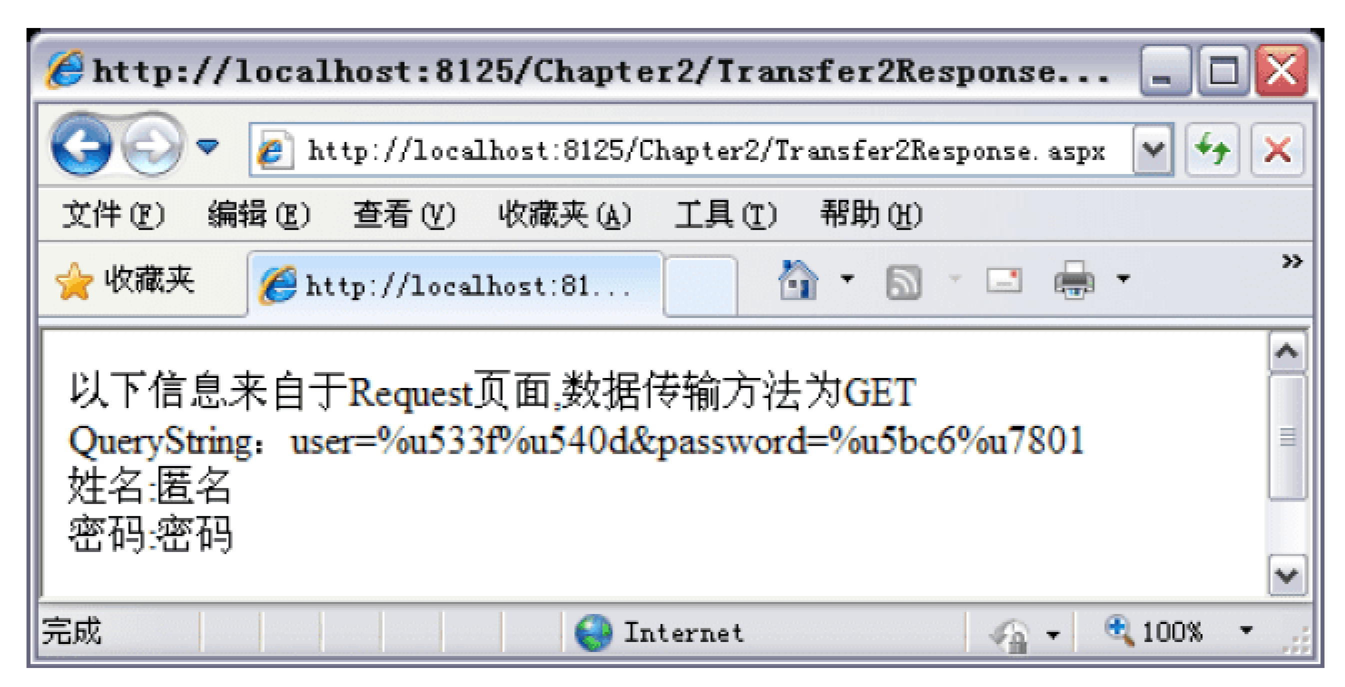


图 2-12　服务器端重定向

从图 2-12 中可以看出，页面已经被重定向到“Response.aspx?user=匿名&password=密码”，然而，浏览器的地址栏并没有改变，仍然显示为 Transfer2Response.aspx，对用户隐藏了新页面名称和查询字符串值。

提示：

Server.Transfer 只能用来重定向到站点内的其他页面，而不能将用户导航到不同域上的页面。

4. 使用 Server 对象

Server 对象的另一个重要功能是对字符进行 URL 和 HTML 的编码和解码。URL 编码的目的是保证所有浏览器能够正确地传输 URL 路径，一些特殊字符如？、&、/、空格和中文字符等，在传输时都有可能让浏览器发生错误。因此，可以先通过编码再将其传输，在需要使用时又通过解码将其还原。HTML 编码的作用是将所有字符全部转换为 HTML 中能够用来显示的字符，例如，<p>如果直接显示就是一个段落，而转换以后就会变成<p>；浏览时就可以正确显示出<p>，从而不会造成错误。

例 2-5：使用 Server 对象获取服务器信息。

(1) 启动 VWD 2010，选择“文件”|“打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 Server.aspx 的 Web 窗体。

(3) 在 Default.aspx.cs 文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("使用 Server 对象获取服务器信息，如下所示：");
    Response.Write("<br>服务器名称：" + Server.MachineName);
    Response.Write("<br>服务器超时时间：" + Server.ScriptTimeout);
    Response.Write("<br>下面将显示一条水平线<hr>");
```

```
            Response.Write("<br>下面将显示字符"+ Server.HtmlEncode(" <hr>"));
        }
```

(4) 编译并运行程序，在浏览器中查看 Server.aspx，效果如图 2-13 所示。

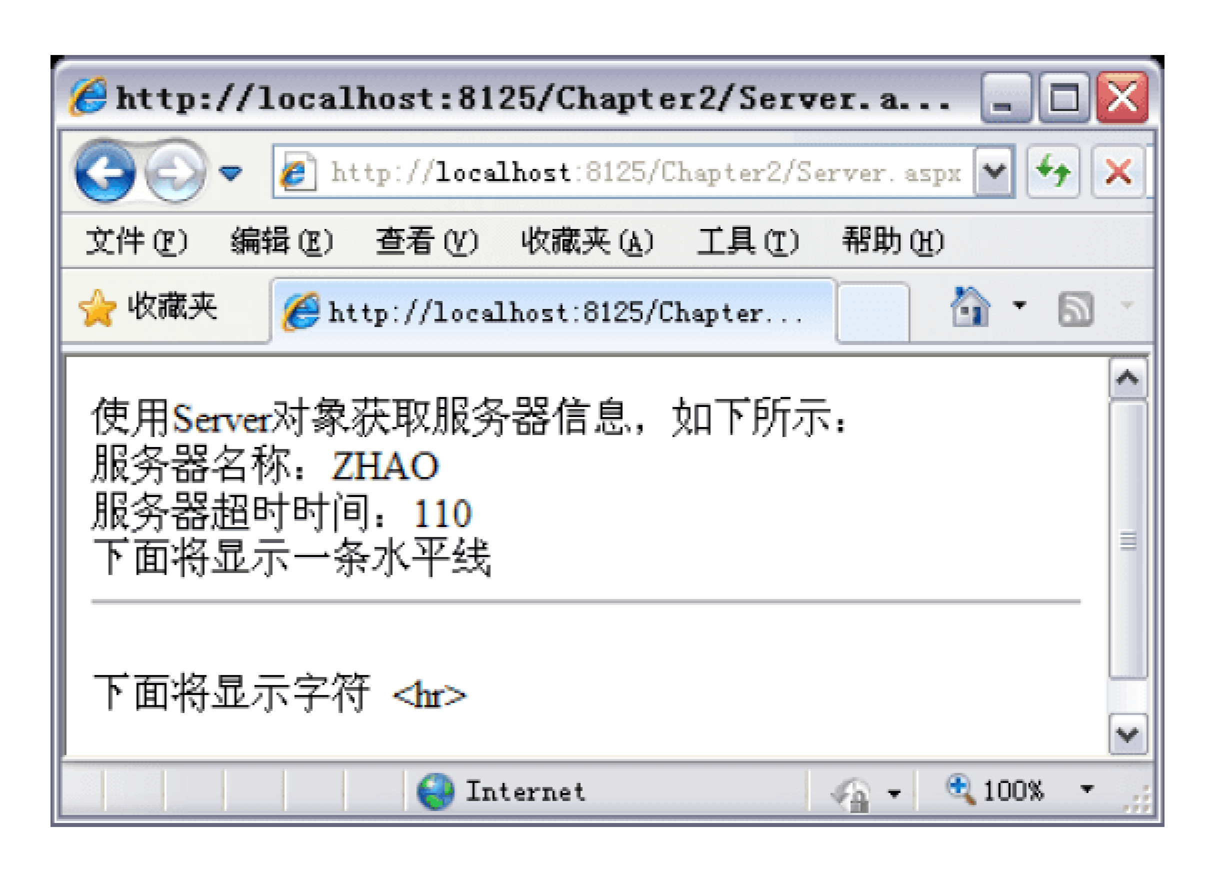


图 2-13　Server 对象示例

提示：

在 Response.aspx 页面中，输出的 QueryString 集合就是经过 URL 编码后的格式，读者可以自己将其用 Server.UrlDecode 方法进行解码，查看解码后的输出效果。

2.2.6　Session 对象

Session 对象是 System.Web.HttpSessionState 类的实例，其作用是储存特定的信息，但是它和 Application 对象储存信息所使用的对象是完全不同的。Application 对象储存的是共享信息，而 Session 储存的信息是局部的，是随用户不同而不同的。如果只需在不同页中共享数据，而不是在不同的客户端之间共享数据就可以使用 Session 对象。

Session 的生命周期是有限的(默认值为 20 分钟)，可以使用 Timeout 属性进行设置。在 Session 的生命周期内，Session 的值是有效的。如果用户在大于生命周期的时间里没有再访问应用程序，Session 就会自动过期，即 Session 对象将会被 CLR 释放，其中保存的数据信息也将丢失。

1. Session 对象的属性

Session 对象提供了对会话状态值以及会话级别设置和生存管理方法的访问，其常用的属性如表 2-15 所示。

表 2-15　Session 对象的常用属性

属　性	描　述
CodePage	获取或设置字符集标识
Contents	获取当前 Session 状态对象的引用

(续表)

属　　性	描　　述
CookieMode	获取当前的 Cookie 模式，以确定系统是否要将 Session 配置为不需要 Cookie 支持
Count	Session 状态集合的总数
IsCookieless	是否需要 Cookie 支持，如果需要就可以将 Session ID 保存在 Cookie 中，如果不需要就必须嵌入在 URL 中
IsNewSession	标志当前 Session 是否是新的 Session
IsReadOnly	是否只读
IsSynchronized	是否同步
Item	通过索引获取或者设置单个 Session 值
Keys	获取 Session 集合的所有键
LCID	获取和设置当前 Session 的本地标识符
Mode	获取当前的 Session 模式
SessionID	获取 Session 的唯一编号，为了区别不同的会话，系统会为每一个会话分配一个唯一的 ID
StaticObjects	获取在 Global.asax 中以<object Runat="Server" Scope="Session" />声明的对象集合
Timeout	获取和设置会话超时时间，如果客户端在连续一个时间段内没有反应，就自动清除会话，断开连接，Timeout 就是这个时间段

2. Session 对象的方法

Session 对象的常用方法如表 2-16 所示。

表 2-16　Session 对象的常用方法

方　　法	描　　述
Add	添加一个新对象到 HttpSessionState 集合
Clear	清除 HttpSessionState 集合中的所有对象
Get	通过索引和名字获得 HttpSessionState 对象
Abandon	清除当前会话
CopyTo	复制 Session 状态集合到一个一维数组
Remove	从 HttpSessionState 集合删除一个对象
RemoveAll	删除 HttpSessionState 集合所有对象
RemoveAt	根据索引删除一个 HttpSessionState 对象

3. Session 变量

每一个 Session 对象都具有唯一的 SessionID 编号，在浏览 ASP.NET 应用程序的整个过程中，都可以存取 Session 对象建立的变量。

技巧：

在 ASP.NET Web 应用程序中可以使用 SessionID 编号判断用户是否仍在 Session 时间，它是直到 Session 对象的 TimeOut 属性设定时间到时或执行了 Abandon 方法后才会结束 Session 时间。

如果同时有多位用户浏览网站，那么每位用户都被指定不同的 SessionID 编号，只允许拥有此 ID 的用户存储此 ID 的 Session 变量。

Session 变量是用户的专用数据，虽然每位用户的 Session 变量名是相同的，但是，变量的值确可能不同。而且每个用户只能存取自己的 Session 变量。例如，用户 liuxf 访问网站，程序为其建立的 Session 变量如下：

```
Session["username"]="刘秀芬";
Session["city"]="沧州";
```

接着，另一位用户 zhaoyd 也请求访问网站，系统也会为他建立一组 Session 变量，但值确不同，例如：

```
Session["username"]="赵艳铎";
Session["city"]="北京";
```

4. 使用 Session 对象

对于 Web 应用程序的用户数据，或是购物车等个人专属数据，通常都使用 Session 对象进行存储和管理。

例 2-6：使用 Server 对象存储用户数据。

(1) 启动 VWD 2010，选择“文件” | “打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 Session1.aspx 的 Web 窗体。

(3) 打开 Session1.aspx 的设计视图，添加一个 TextBox 控件和一个 Button 控件到 Web 窗体中，系统自动为它们命名为 TextBox1 和 Button1。

(4) 在后台代码文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("下面是获取到的 Session 信息：");
    Response.Write("<br>SessionID：" + Session.SessionID);
    Response.Write("<br>Session 的数量：" + Session.Count);
    Response.Write("<br>Session 的模式：" + Session.Mode);
    Response.Write("<br>Session 的有效期" + Session.Timeout);
}
```

(5) 为按钮控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Session["username"] = TextBox1.Text;
```

```
        Response.Redirect("Session2.aspx");
    }
```

(6) 添加一个名为 Session2.aspx 的 Web 窗体，在 Session2.aspx.cs 的 Load 事件处理程序中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["username"] != null)
    {
        Response.Write("通过 Session 对象获取到的信息：" + Session["username"]);
        Session.Remove("username");
    }
    else
    {
        Response.Redirect("Session1.aspx");
    }
}
```

(7) 编译并运行程序，在浏览器中请求 Session1.aspx 页面，如图 2-14 所示，在文本框中输入信息，然后单击 Button 按钮，效果如图 2-15 所示。

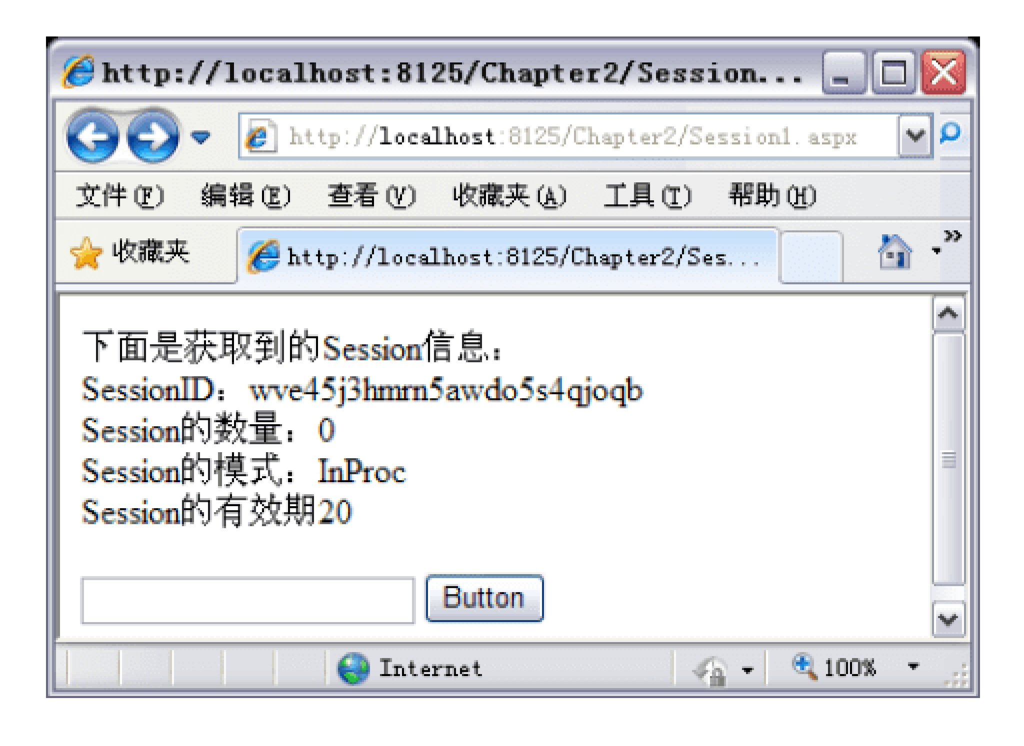


图 2-14　Session 对象示例

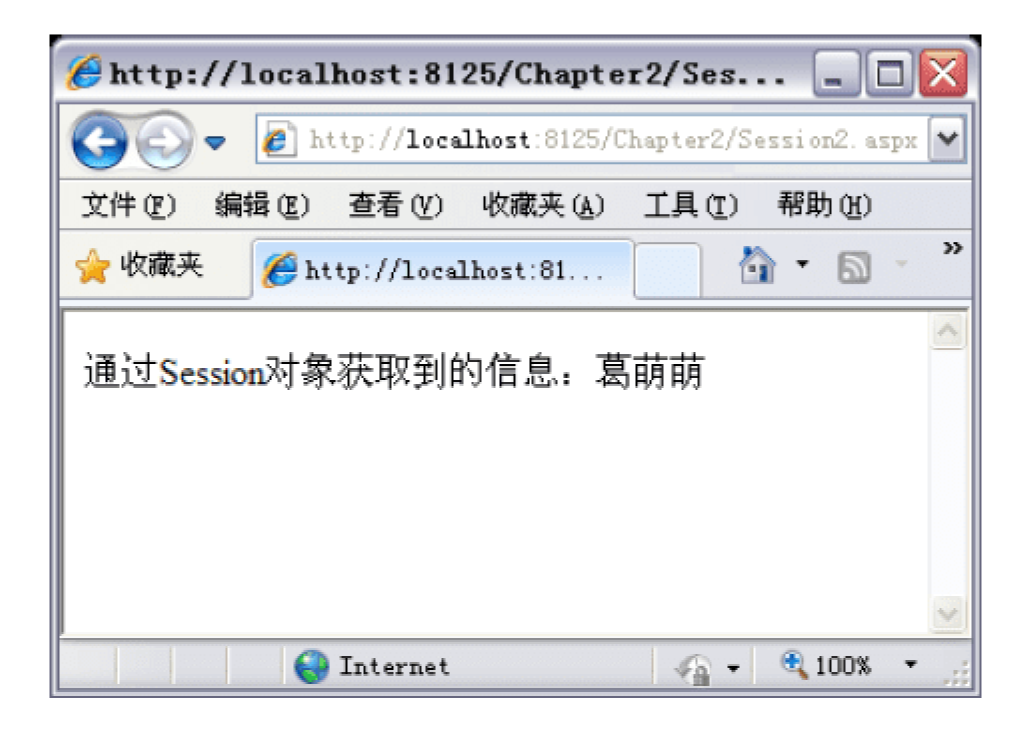


图 2-15　通过 Session 对象获取信息

2.2.7　Cookie 对象

Cookie 是比较常用的一种对象，通常用来存储少量浏览者的信息，如浏览者的喜好、用户名和 Email 地址等信息，以便于当浏览者再次登录网站时，不必再次填写这些信息。

1. Cookie 对象简介

Cookie 俗称“小甜饼”，它其实只是一些小文本，将一些用户信息储存在客户端的机器中，它全部存储于 Windows 目录下的 Cookie 文件夹中，以便于在每次请求时被服务器在设定的时期内进行读取。Cookie 的储存大小是有限制的，一般浏览器会将其大小控制在 4096 个字节以内。

注意：

Cookie 与网站关联，而不是与特定的页面关联。因此，无论用户请求站点中的哪一个页面，浏览器和服务器都将交换 Cookie 信息。用户访问不同站点时，各个站点都可能会向用户的浏览器发送一个 Cookie，浏览器会分别存储所有 Cookie。

当用户请求站点中的页面时，应用程序发送给该用户的不仅仅是一个页面，还有一个包含日期和时间的 Cookie，用户的浏览器在获得页面的同时也获得了该 Cookie，并将它存储在用户硬盘上的文件夹中。以后，如果该用户再次请求站点中的页面，当该用户输入 URL 时，浏览器便会在本地硬盘上查找与该 URL 关联的 Cookie。例如，当用户登录 163 的邮箱后，如果在 Cookie 中记录了用户名信息，那么在 Cookie 信息失效以前，该用户在同一台计算机再次登录时就无须提供用户名。

使用 Cookie 具有如下优点。

- 可配置到期规则：Cookie 可以在浏览器会话结束时到期，或者可以在客户端计算机上无限期存在，这取决于客户端的到期规则。
- 不需要任何服务器资源：Cookie 存储在客户端并在发送后由服务器读取。
- 简单性：Cookie 是一种基于文本的轻量结构，包含简单的键/值对。
- 数据持久性：虽然客户端计算机上 Cookie 的持续时间取决于客户端上的 Cookie 过期处理和用户干预，但 Cookie 通常是客户端上持续时间最长的数据保留形式。

2. Cookie 的修改与删除

浏览器向服务器发出请求时，会随请求一起发送该服务器的 Cookie。在 ASP.NET 应用程序中，是使用 Request 和 Response 对象的 Cookies 集合对象来读取和设置 Cookie 的。

主要的操作有 3 种，即设置 Cookie、删除 Cookie 和获取 Cookie 的内容。

服务器是不能直接修改 Cookie 的，修改 Cookie 的方法是重新创建一个同名的、具有新值的新 Cookie。例如：

```
HttpCookie cookie=new HttpCookie("name")
Cookie.Value="葛萌萌"
```

上述代码创建了一个名为 name 的 HttpCookie 实例。然后通过设置 Cookie 的 Value 属性为其设置一个值，从而将源 Cookie 中的 name 修改为“葛萌萌”。

在一个 Cookie 中可以存储一个值，也可以储存多个值。通过 Value 属性，可以在 Cookie 中存储一个值，如上面的代码；通过 Cookie 的 Values 集合，可以在同一个 Cookie 中储存多个值。例如：

```
HttpCookie cookie=new HttpCookie("student");
Cookie.Values.Add("Admin", "赵飞燕");
Cookie.Values.Add("Member1", "金百合");
Cookie.Values.Add("Member2", "小石头");
```

Values 集合的 Add 方法中第一个参数为关键字(Key)，第二个参数是设置的值(Value)。

由于 Cookie 在用户计算机中，因此无法通过编程将其直接移除。但是，可以让浏览器来删除 Cookie，具体做法是创建一个与要删除的 Cookie 同名的新 Cookie，并将该 Cookie 的到期日期设置为过去的某个日期，当浏览器检查 Cookie 的到期日期时，便会丢弃这个已过期的 Cookie。

3. 确定浏览器是否接受 Cookie

除了限制 Cookie 的大小，浏览器还限制站点可以在用户计算机上存储的 Cookie 的数量。大多数浏览器只允许每个站点存储 20 个 Cookie，如果试图存储更多的 Cookie，则存放最早的 Cookie 便会被覆盖掉。有些浏览器还会对它们将接受的来自所有站点的 Cookie 总数做出绝对限制，通常为 300 个。

另外，用户还可以将自己的浏览器设置为拒绝接受 Cookie。设置为拒绝接受 Cookie 后，虽然不能向客户端写入 Cookie 信息，但是不会引发任何错误。同样，浏览器也不向服务器发送有关其当前 Cookie 设置的任何信息。确定客户端浏览器是否接受 Cookie 的一种方法是尝试编写一个 Cookie，然后再读取该 Cookie。如果无法读取已编写的 Cookie，则可以假定浏览器不接受 Cookie。

4. 使用 Cookie 对象

例 2-7 将演示 Cookie 的使用方法。

例 2-7：演示 Cookie 的设置与读取。

(1) 启动 VWD 2010，选择“文件” | “打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 Cookie.aspx 的 Web 窗体。

(3) 打开 Cookie.aspx 的设计视图，添加一个 Label 控件、一个 TextBox 控件和一个 Button 控件到 Web 窗体中，修改 Button 控件的 Text 属性为“提交”，同时在控件的旁边输入一些提示性的文本信息，页面的设计效果如图 2-16 所示。

(4) 在后台代码文件中添加页面的 Load 事件和按钮的单击事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = Request.Cookies["name"];
    if (cookie == null)
        Label1.Text = "无 Cookie 信息";
    else
        Label1.Text = cookie.Value ;
}
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text == "")
    {
        Label1.Text = "请在下面的文本框中输入新的 Cookie 值!";
        return;
    }
```

```
            HttpCookie cookie = Request.Cookies["user"];
            if (cookie == null)
                cookie = new HttpCookie("name");
            cookie.Value = TextBox1.Text;
            cookie.Expires = DateTime.Now.AddDays(7);
            Response.Cookies.Add(cookie);
        }
```

(5) 编译并运行程序，在浏览器中打开 Cookie.aspx 页面，初次访问该页面时，由于 Cookie 中没有值，所以显示效果如图 2-17 所示。

(6) 在文本框中输入一个新值，单击“提交”按钮，将该值保存到 Cookie 中。

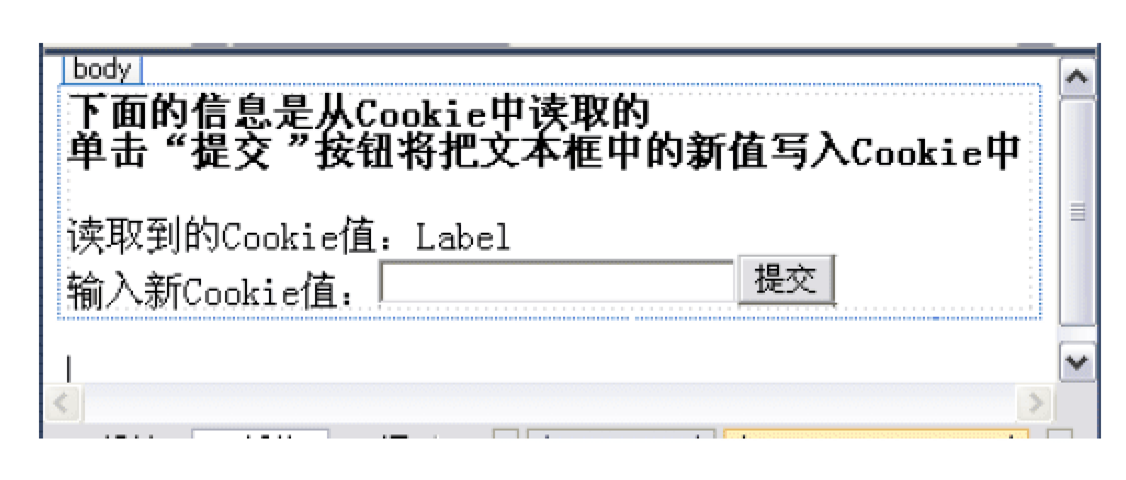


图 2-16　页面设计与布局



图 2-17　页面初始效果图

(7) 按 F5 键刷新页面，可以看到页面发生了变化，刚才保存的 Cookie 值被读取出来了，如图 2-18 所示。

(8) 接下来，在浏览器窗口中选择“工具”|“Internet 选项”命令，打开“Internet 选项”对话框，如图 2-19 所示。

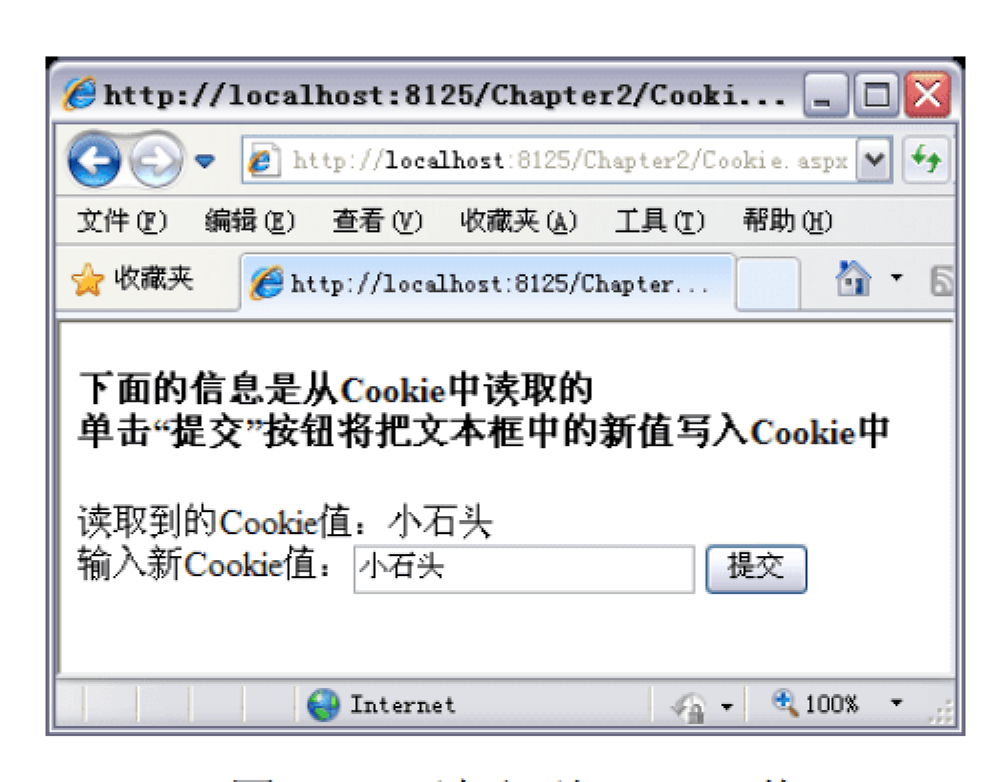


图 2-18　读取到 Cookie 值

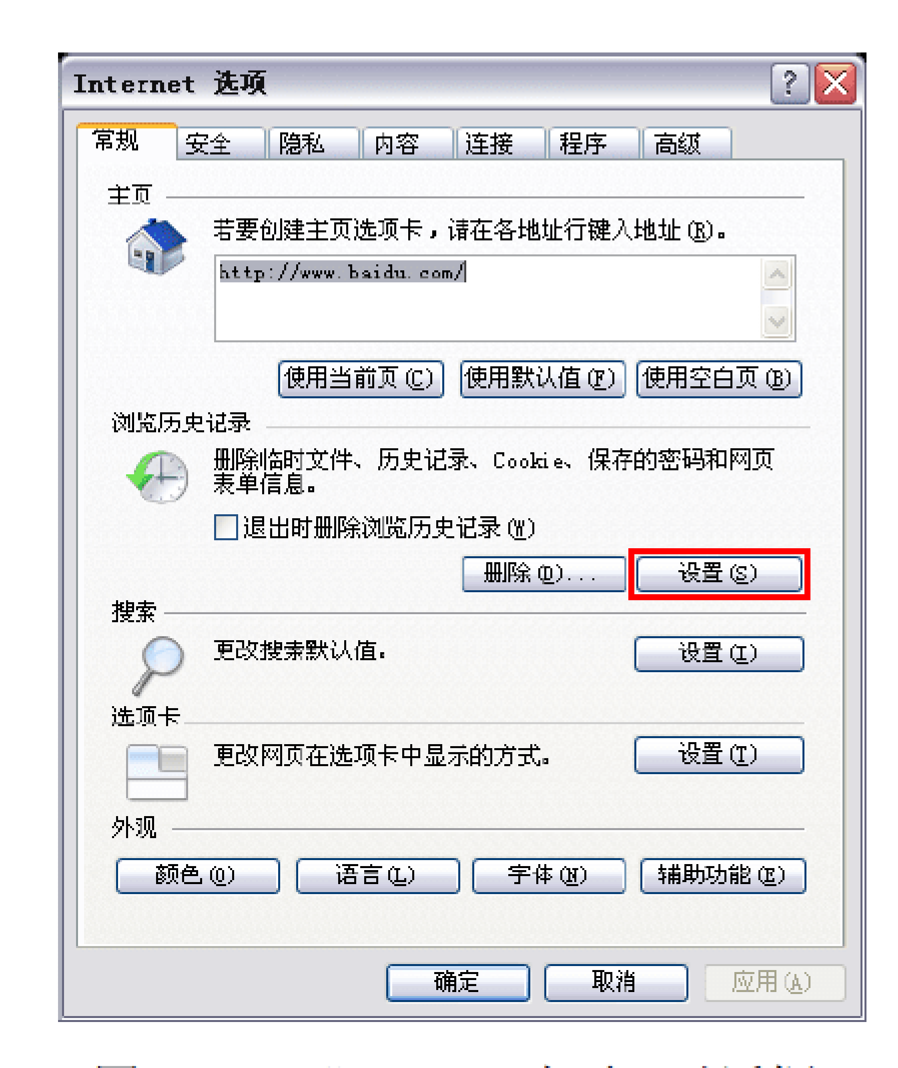


图 2-19　“Internet 选项”对话框

(9) 单击“浏览历史记录”选项区域中的“设置”按钮，可以打开“Internet 临时文件和历史记录设置”对话框，如图 2-20 所示。

(10) 单击“查看文件”按钮，将打开 Windows 资源管理器，同时定位到 Cookie 存放的目录，如图 2-21 所示，此目录中有很多文本文件和一些图片文件等 Internet 临时文件。

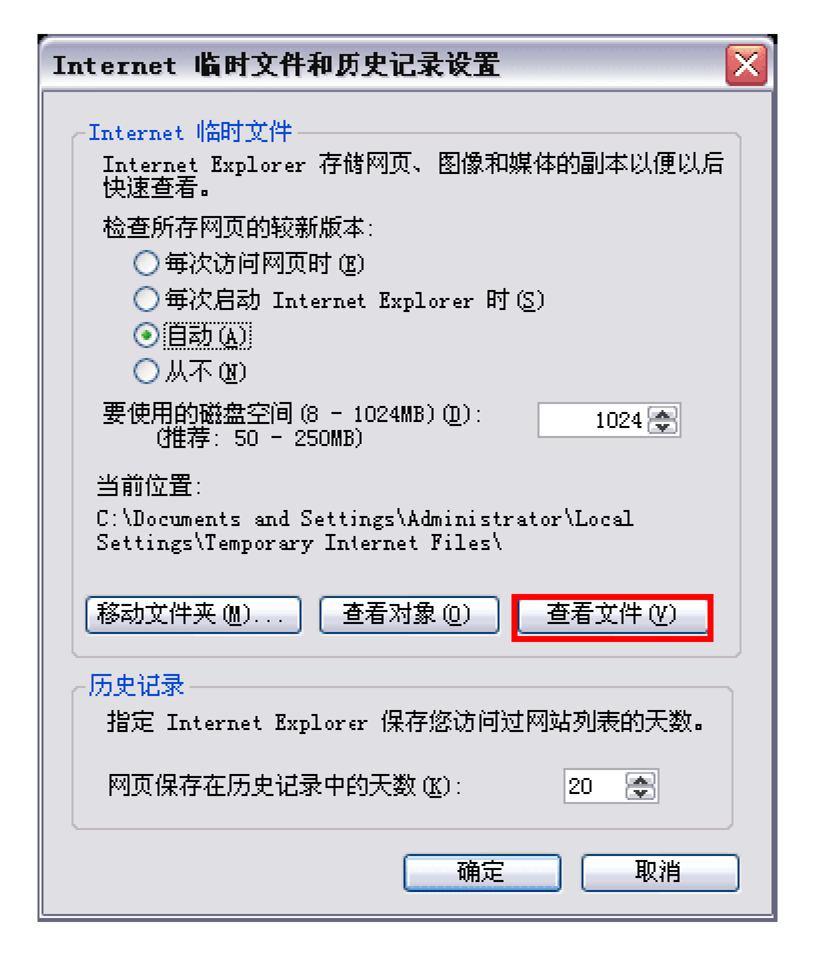


图 2-20　“Internet 临时文件和历史记录设置”对话框

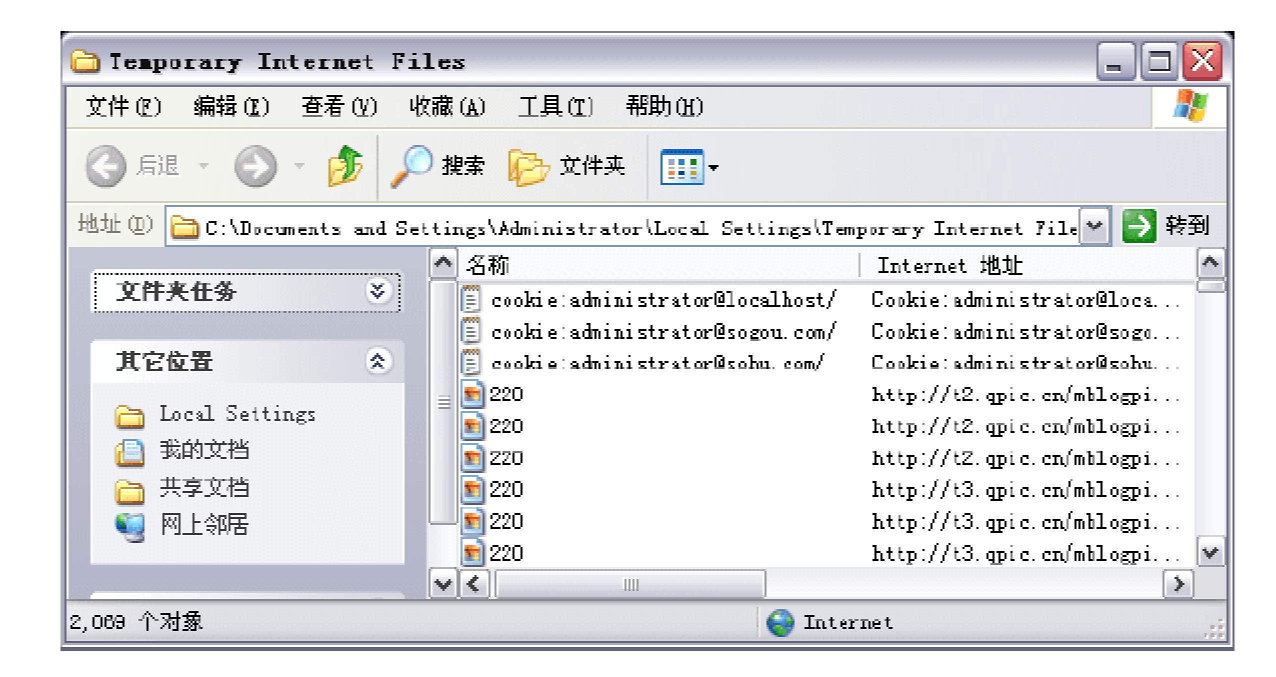


图 2-21　打开 Cookie 存放的目录

(11) Cookie 文件一般都是以 cookie 打头，以网站域名为结尾的文件，如本例中网站的 Cookie 文件在笔者的电脑上是 cookie:administrator@localhost/。

(12) 双击 Cookie 文件，可以通过文本编辑器打开并查看该文件，如图 2-22 所示。

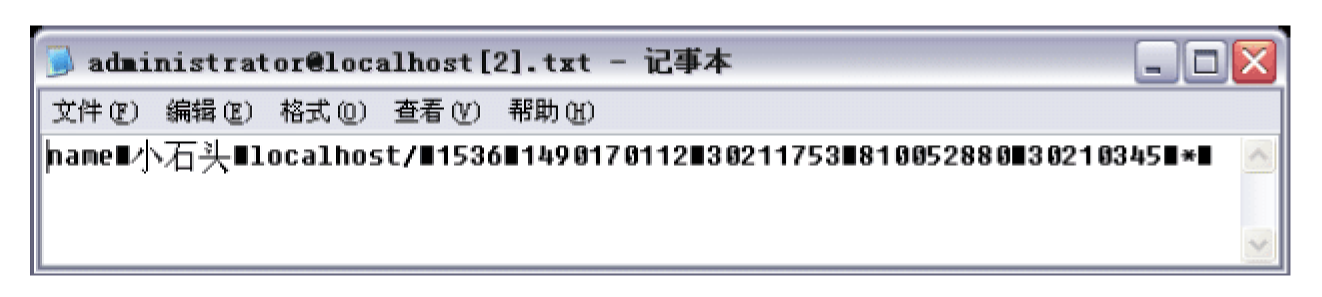


图 2-22　查看 Cookie 文件

2.2.8　ViewState 对象

ViewState(视图状态)对象是 Page 对象的一个属性，是状态管理中常用的一种对象，可以用来保存页和控件的值。

说明：

视图状态中存储的常见数据类型有以下几种，分别是字符串、整数、布尔值、Array 对象、ArrayList 对象、哈希表和泛型对象等。

1. ViewState 简介

视图状态是 ASP.NET 页框架默认情况下用于保存往返过程之间的页面信息以及控件值的方法。当呈现页的 HTML 形式时，需要在回发过程中保留的页的当前状态和值将被序列化为 Base64 编码的字符串，并输出到视图状态的隐藏字段中。通过实现自定义的 PageStatePersiste 类以存储页数据，也可以更改默认行为并将视图状态存储到另一个位置，如 SQL Server 数据库。

程序员可以通过使用页面的 ViewState 属性将往返过程中的数据保存到 Web 服务器端，然后利用自己的代码访问视图状态。ViewState 属性是 StateBag 类的实例，它是一个包含键/值对的字典，并通过唯一的键名来访问对应的值。

使用 ViewState 可以带来很多方便，但是也有一些需要注意的问题。

- 视图状态提供了特定 ASP.NET 页面的状态信息。如果需要在多个页上使用信息，或者需要在访问网站时保留信息，则应当使用另一个方法(如应用程序状态、会话状态或个性化设置)来维护状态。
- 视图状态信息将序列化为 XML，然后进行 Base64 编码，这将生成大量的数据。将页回发到服务器时，如果视图状态包含大量信息，则会影响页的性能。
- 虽然使用视图状态可以保存页和控件的值，但是在某些情况下，需要关闭视图状态。如使用 GridView 控件显示数据，单击 GridView 控件的下一页按钮，此时，GridView 控件呈现的数据已经不再是前一页的数据，那么如果使用视图状态将前一页数据保存下来，不仅没有必要而且还会生成大量隐藏字段，增大页面的体积。

如果隐藏字段中的数据量过大，某些代理的防火墙将禁止访问包含这些数据的页。由于所允许的最大数据量随所采用的防火墙和代理的不同而不同，因此大量隐藏字段可能会导致偶发性问题。为了帮助避免这一问题，如果 ViewState 属性中存储的数据量超过了页的 MaxPageStateFieldLength 属性中指定的值，该页会将视图状态拆分为多个隐藏字段，可以使每个单独字段的大小在防火墙拒绝的大小以下。

2. 如何关闭 ViewState

并不是所有控件都一直依赖于 View State。有很多控件能维持它们自己的某些状态。这些控件包括 TextBox、CheckBox、RadioButton 和 DropDownList。它们能维持它们的值，这是因为它们在浏览器中被呈现为标准的 HTML 表单控件。例如，TextBox 服务器控件在浏览器中看起来如下所示：

```
<input name="TextBox1" type="text" value="Initial Text" id="TextBox1" />
```

当发送回一个带有这样的 TextBox 的页面时，浏览器也会将控件的值发送回服务器。然后 ASP.NET 运行库就能再次用这个值来预先填写文本框，而不需要从 ViewState 中获取值。显然，这也比将值存储在 ViewState 中更有效。如果这些值也存储在 ViewState 中，值就会被发送到服务器中两次：一次在文本框中，另一次在 View State 中。当值比较大时，就会大大增加页面的大小，增加页面加载的时间。因此，在不需要时最好关闭它。这样就能最小化隐藏字段_VIEWSTATE 的大小。

关闭 ViewState 很容易，可以在以下 3 个地方做到。

- 在 Web 站点级别

可以在根站点的 web.config 文件(2.3 节将详细介绍 web.config 文件)中通过修改<system.web>下面的<pages>元素，将 enableViewState 特性设置为 false 来完成。

```
<pages enableViewState="false">
   ...
</pages>
```

通常不在站点级别关闭 ViewState，因为在站点级别关闭 ViewState 后，将无法为特定的控件打开这个功能。幸运的是，ASP.NET 4.0 提供了一个新的属性 ViewStateMode，它提供了关于 ViewState 如何使用的更多控制。

- 在页面级别

在每个页面的页面指令中，可以将 EnableViewState 设置为 False，例如：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default"   EnableViewState="False" %>
```

对于确认根本不需要 View State 的页面来说，这种方法是非常有用的。

- 在控件级别

各个 ASP.NET 服务器控件允许分别设置 EnableViewState 属性，这样可以选择关闭某些控件，而使其他控件保持打开。

一旦在更高级别(web.config 或页面级别)上关闭了 ViewState，就不能再在一个低层级别(页面或特定控件级别)上打开这个功能。但是，使用新的 ViewStateMode 属性仍能完成如下工作。

(1) 禁止在 web.config 文件中关闭 View State。

(2) 在页面级别，将 EnableViewState 设置为 True，将 ViewStateMode 设置为 Disabled，如下所示：

```
<%@ Page Language="C#"...EnableViewState="True" ViewStateMode="Disabled" %>
```

上述代码可以关闭页面中所有控件的 ViewState，除了那些再次明确地将 ViewStateMode 属性设置为 Enabled 的控件以外。

(3) 如果想让控件支持 ViewState，可以将控件的 ViewStateMode 属性设置为 Enabled，例如：

```
<asp:Label ID="Label1" runat="server" Text="Label" ViewStateMode="Enabled" />
```

3. 使用 ViewState

例 2-8 将演示 ViewState 的使用。

例 2-8：使用 ViewState 对象。

(1) 启动 VWD 2010，选择“文件”|“打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 ViewState.aspx 的 Web 窗体。

(3) 打开 ViewState.aspx 的设计视图，添加三个 Label 控件、两个 TextBox 控件和两个 Button 控件到 Web 窗体中，控件的 Text 属性和页面布局如图 2-23 所示。

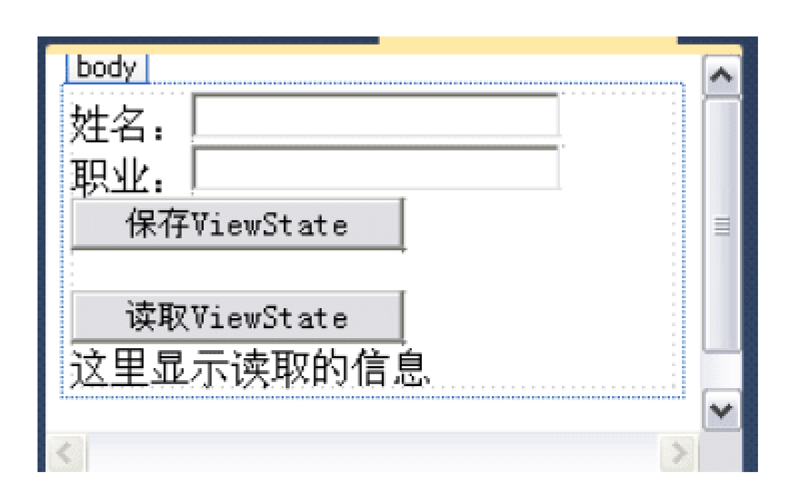


图 2-23 页面布局及控件的 Text 属性

(4) 在页面的后台代码文件中添加页面的 Load 事件和两个按钮的单击事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        ViewState.Add("user", "赵艳铎");
        ViewState.Add("vocation", "工程师");
    }
}
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox2.Text != "")
        ViewState["user"] = TextBox1.Text;
    if (TextBox2.Text != "")
        ViewState["vocation"] = TextBox2.Text;
}
protected void Button2_Click(object sender, EventArgs e)
{
    Label3.Text = "ViewState 信息如下:<br>姓名：";
    Label3.Text += ViewState["user"];
    Label3.Text += "<br>职业：";
    Label3.Text += ViewState["vocation"];
}
```

(5) 编译并运行程序，在浏览器中打开 ViewState.aspx 页面。单击“读取 ViewState”按钮，读取 ViewState 初值，如图 2-24 所示；输入姓名和职业后，单击“保存 ViewState”按钮，然后再次单击“读取 ViewState”按钮，读取 ViewState 的新值，效果如图 2-25 所示。

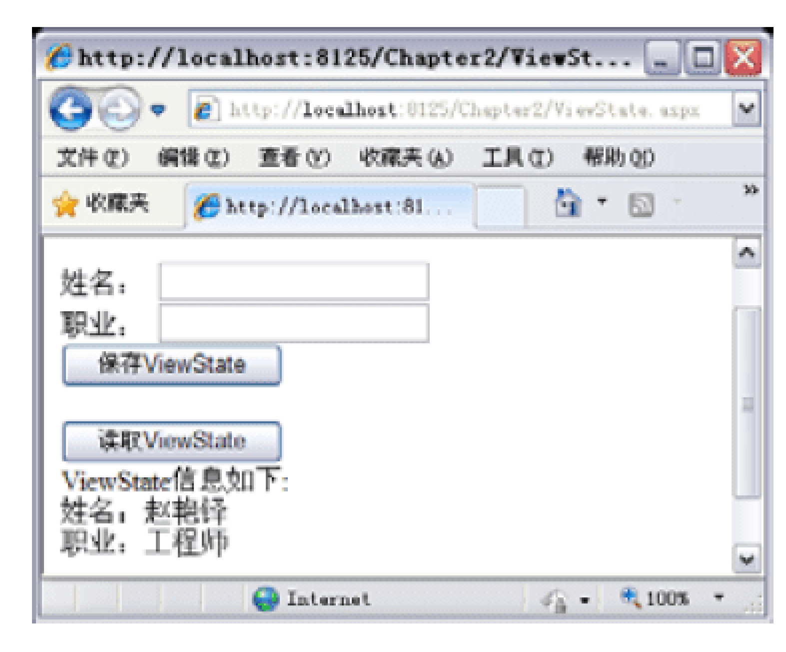


图 2-24 读取视图状态初值

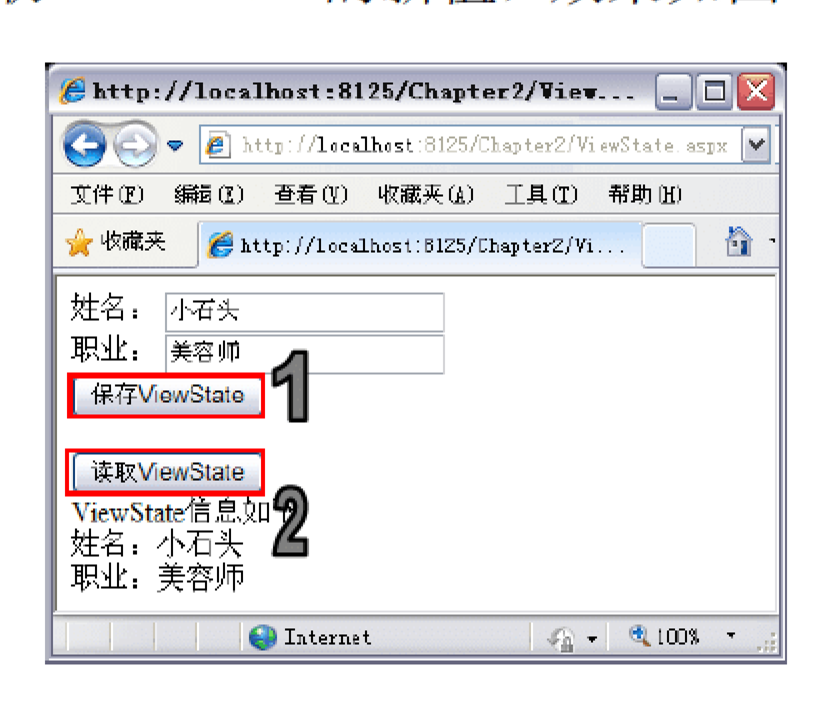


图 2-25 读取 ViewState 中的新值

2.3　ASP.NET 配置管理

使用 ASP.NET 配置系统的功能，可以配置整个服务器上的所有 ASP.NET 应用程序、单个 ASP.NET 应用程序和各个页面或应用程序子目录，也可以配置各种具体的功能，如身份验证模式、页缓存、编译器选项、自定义错误、调试和跟踪选项等。

2.3.1　web.config 文件

每一个 Web 应用程序都包含一个 web.config 配置文件，该配置文件为 ASP.NET 提供了各种基础的设置。

web.config 是一份 XML 文件，配置文件的全部内容都嵌套在根元素<configuration>中。内含 Web 应用程序相关设定的 XML 标记，可用来简化 ASP.NET 应用程序的相关设定。

web.config 文件位于 Web 应用程序的任何目录中，统一命名为 web.config，它决定了所在目录及其子目录的配置信息，并且子目录下的配置信息会覆盖其父目录的配置，即子目录如果没有 web.config 文件，就继承父目录 web.config 文件的相关设定；如果子目录有 web.config 文件，就会覆盖父目录 web.config 文件中的相关设定。在运行状态下，ASP.NET 会根据远程 URL 请求，把访问路径下的各个 web.config 配置文件叠加，产生一个唯一的配置集合。

举例来说，对 URL：http://localhost/website/ownconfig/test.aspx 的访问，ASP.NET 会根据以下顺序来决定最终的配置情况。

- .\Microsoft.NET\Framework\{version}\web.config(默认配置文件)
- .\webapp\web.config(应用的配置)
- .\webapp\ownconfig\web.config(自己的配置)

1. 配置文件的语法规则

web.config 的全部内容都被置于标记<configuration>和</configuration>之间。在 XML 标记的属性就是设定值，标记名称和属性值格式是字符串，第一个开头字母是小写，之后每一个字的首字母大写，例如<appSetting>。Web 配置文件的范例如下所示。

```
<configuration>
    <appSettings>
      <add key="dbType" value="Access Database"/>
    </appsettings>
    <connectionsStrings>
      <add name="provider" connectionString="Microsoft.Jet.OLEDB.4.0;"/>
      <add name="database" connectionString="/chapter5/student.mdb"/>
    </connectionsStrings>
    <system.web>
      <sessionState cookieless="false" timeout="10"/>
```

```
        <compilation defaultLanguage="C#" debug="true"/>
        <globalization   fileEncoding="gb2312"   requestEncoding="gb2312" culture= "zh-CN"/>
        <customErrors mode="RemoteOnly"/>
    </system.web>
</configuration>
```

可以看到，这段配置信息是一个基于 XML 格式的文件，根标记是<configuration>，所有的配置信息均被包括在<configuration>及</configuration>标签中间，其子标记<appSettings>、<connectionsStrings>和<system.web>是各设定区段。在<system.web>下的设定区段属于 ASP.NET 相关设定，常用的区段标记如表 2-17 所示。

表 2-17　常用设定区段标记说明

设 定 区 段	描　　述
<anonymousIdentification>	控制 Web 应用程序的匿名用户
<authentication>	设定 ASP.NET 的验证方式
<authorization>	设定 ASP.NET 用户授权
<browserCaps>	设定浏览程序兼容组件 HttpBrowserCapabilities
<compilation>	设定 ASP.NET 应用程序的编译方式
<customErrors>	设定 ASP.NET 应用程序的自动错误处理
<globalizations>	关于 ASP.NET 应用程序的全球化设定，也就是本地化设定
<httpHandlers>	设定 HTTP 处理是对应到 URL 请求的 HttpHandler 类
<httpModules>	创建、删除或清除 ASP.NET 应用程序的 HTTP 模块
<httpRuntime>	设定 ASP.NET HTTP 运行时的配置，这些配置决定如何处理对 ASP.NET 应用程序的请求
<machineKey>	设定在使用窗体基础验证的 Cookie 数据时，用来加码和解码的金钥匙
<membership>	设定 ASP.NET 的 Membership 机制
<pages>	设定 ASP.NET 程序的相关设定，即 Page 指引命令的属性
<profile>	设定个人化信息的 Profile 对象
<roles>	设定 ASP.NET 的角色管理
<sessionState>	设定 ASP.NET 应用程序的 Session 状态 HttpModule
<siteMap>	设定 ASP.NET 网站导航系统
<webParts>	设定 ASP.NET 应用程序的网页组件
<webServices>	设定 ASP.NET 的 Web 服务

2. 在<appSettings>中创建用户变量

在 Web 配置文件的<appSettings>区段可以创建 ASP.NET 程序所需的参数，每个<add>标记可以创建一个参数，属性 key 是参数名称，value 是参数值。ASP.NET 2.0 新增了

<connectionStrings>区段，可以指定数据库连接字符串，在<connectionStrings>标记的<add>子标记也可以创建数据库连接字符串，属性 name 是名称，connectionStrings 是连接字符串的值。

说明：

<connectionStrings>标记的使用将在第 5 章介绍。下面举例说明<appSettings>中参数的创建与使用。

在 Web 配置文件的<appSettings>区段通过<add>标记创建的参数，在程序中可以使用 System.Web.Configuration 命名空间的 WebConfigurationManager 类来获取。

例 2-9：在 web.config 的<appSettings>区段中创建和使用用户变量。

(1) 启动 VWD 2010，选择“文件”|“打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加一个名为 AppSettings.aspx 的 Web 窗体。在<form>标记上方添加如下提示性文本：

```
<h3>读取 web.config 中<% =Server.HtmlEncode("<appSettings>"). %>中配置的用户变量</h3>
```

(3) 切换到 AppSettings.aspx 的设计视图，添加一个 Label 控件到 Web 窗体中。

(4) 打开 web.config 文件，在<appSettings>配置子节点中添加一个变量，代码如下：

```
<appSettings>
    <add key="MyParam" value="金百合拉丁舞培训学校"/>
</appSettings>
```

技巧：

如果 web.config 文件中没有<appSettings>标记，则可以自行添加，将其包含在<configuration>标签下即可。

(5) 打开后台代码文件 AppSettings.aspx.cs，首先在上方添加 using 语句，引入所需的命名空间：

```
using System.Web.Configuration;
```

(6) 接着，添加页面的 Load 事件处理程序，代码如下：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "用户变量的值为：" +
WebConfigurationManager.AppSettings["MyParam"];
    }
```

(7) 编译并运行程序，在浏览器中打开 AppSettings.aspx 页面，如图 2-26 所示。



图 2-26　页面运行效果

3. 在 sessionState 区段设置 Session 状态

ASP.NET 的 Session 状态管理拥有扩展性，可以在 web.config 文件的<sessionState>区段设定 Session 状态管理，该区段属于<system.web>子标记。其常用属性如表 2-18 所示。

表 2-18　<sessionState>区段的常用属性

属　性	描　述
mode	Session 状态存储的模式，可以是 off(不存储)、InProc(使用 Cookie)、StateServer(使用状态服务器)和 SqlServer(存储在 SQL Server 中)
cookieless	是否使用 Cookie 存储 Session 状态。True 表示不使用，False 表示使用
timeout	Session 时间的期限，以分钟计算，默认为 20 分钟，与 Session 对象的 TimeOut 属性功能相同

2.3.2　Global.asax 文件

作为网络应用程序，在执行之前有时需要初始化一些重要的变量，而且这些工作必须发生在所有程序执行之前，Global.asax 文件便是为此目的而设计的。

Global.asax 是 ASP.NET 应用程序的“全局应用程序类”，该文件是应用程序用来保持应用程序级的事件、对象和变量的。一个 ASP.NET 应用程序只能有一个 Global.asax 文件。

注意：

Global.asax 存放的位置是固定的，必须存放在当前应用所在的虚拟目录的根目录下。如果放在虚拟目录的子目录中，Global.asax 文件将不会起任何作用。

1. 创建 Global.asax 文件

Global.asax 文件是 Web 应用程序的系统文件，属于选项文件，可有可无。当需要使用 Application 和 Session 对象的事件处理程序时，就需要创建此文件。

在 VWD 2010 中通过“添加新项”对话框就可以创建 Global.asax 文件，在“模板”选项中选择“全局应用程序类”选项即可。

说明：

如果网站已经创建了 Global.asax 文件，则“添加新项”对话框中就看不到“全局应用程序类”模板了。

Global.asax 文件主要是定义 Web 应用程序的 Application_Start()、Application_End()、Session_Start()和 Session_End()等事件处理程序。按照 VWD 模板添加的 Global.asax 如下所示。

```
<%@ Application Language="C#"%>
<script runat="server">
    void Application_Start(object sender,EventArgs e)
    {
        //在应用程序启动时运行的代码
    }
        void Application_End(object sender, EventArgs e)
    {
        //在应用程序关闭时运行的代码
    }
     void Application_Error(object sender, EventArgs e)
    {
        //在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender, EventArgs e)
    {
        //在新会话启动时运行的代码
    }
</script>
```

在窗体页中，只能处理单个页面的事件，而在 Global.asax 文件中则可以处理整个应用程序中的事件。除了上述代码模板中列举的事件，在 Global.asax 文件中还可以加入其他事件的处理函数。表 2-19 列出了可以在 Global.asax 中处理的事件。

表 2-19　Global.asax 中可以处理的事件

事　件	描　述
Application_AuthenticateRequest	每个请求都会触发该事件，并且可以在此函数中设置自定义的验证
Application_BeginRequest	虽然在 VWD 的代码模板中没有该事件的处理，不过可以在 Global.asax 中添加。该事件是在每个请求到达服务器，并且在处理该请求前，会触发的事件
Application_End	应用程序关闭时触发该事件。该函数很少使用，因为 ASP.NET 可以很好地关闭和清除内存对象
Application_Error	在应用程序中抛出任何错误时都会触发该事件。通常在此函数中提供应用程序级的错误处理或者记录错误事件

(续表)

事　件	描　述
Application_Start	在应用程序接收到第一个请求时调用，通常在此函数中定义应用程序级变量或状态
Session_Start	类似于 Application_Start，不过是针对每个客户端第一次访问应用程序时调用
Session_End	以进程内模式使用会话状态时，如果用户离开应用程序将会触发该事件

与页面指令一样，Global.asax 文件也可以使用应用程序指令，这些指令都可以包含特定于该指令的一个或多个属性/值对。下面列出了 ASP.NET 中支持的应用程序指令。

- @Application：定义 ASP.NET 应用程序编译器所使用的应用程序特定的属性。该指令只能在 Global.asax 文件中使用。
- @Import：显式将命名空间导入到应用程序中。
- @Assembly：在分析时将程序集链接到应用程序。

2. 使用 Global.asax 文件

例 2-10：演示 Global.asax 文件的使用，同时回顾前面介绍的内置对象的使用。

(1) 启动 VWD 2010，选择“文件” | “打开网站”命令，打开网站 Chapter2。

(2) 在网站中添加“全局应用程序类”Global.asax。

(3) 在 Global.asax 文件中添加如下代码：

```
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    {
        // 在应用程序启动时运行的代码
        Application["info"] = "开始 Application...<br/>"
    }
    void Application_End(object sender, EventArgs e)
    {
        //  在应用程序关闭时运行的代码
    }
    void Application_Error(object sender, EventArgs e)
    {
        // 在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender, EventArgs e)
    {
        Response.Write(Application["info"].ToString());
        Application["info"] = "";//清空 Application 变量
        Response.Write("开始 Session...<br/>");
```

```
        }
        void Session_End(object sender, EventArgs e)
        {
            Application["info"] = "结束 Session...<br/>";
        }
        void Application_BeginRequest(object sender, EventArgs e)
        {
            Response.Write("开始 Request...<br/>");
        }
        void Application_EndRequest(object sender, EventArgs e)
        {
            Response.Write("结束 Request...<br/>");
        }
    </script>
```

提示：

此时，再访问本章前面创建的所有页面都将发生变化，页面的顶端和最下端至少会增加“开始 Request …”和“结束 Request …”信息。

(4) 为了更全面地了解各个事件的发生顺序，再添加一个名为 testGlobal.aspx 的 Web 窗体。打开该页面的设计视图，添加两个 Button 控件到 Web 窗体中，Text 属性分别为“刷新”和“结束会话”。

(5) 在 testGlobal.aspx 页面的<body>标记中添加如下代码：

```
    <body>
        页面内容...
        (<% if (Session.IsNewSession)
                Response.Write("新的 Session 时间");
            else
                Response.Write("同一个 Session 时间");
         %>)
        <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" Text="刷新" onclick="Button1_Click" /> 
            <asp:Button ID="Button2" runat="server" Text="结束会话" onclick="Button2_Click" />
        </div>
        </form>
    </body>
```

(6) 在后台代码文件中添加页面的 Load 事件和按钮的单击事件处理程序，代码如下：

```
        protected void Page_Load(object sender, EventArgs e)
        {
            Response.Write("加载页面...<br/>");
        }
```

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("刷新页面...<br/>");
}
protected void Button2_Click(object sender, EventArgs e)
{
    Session.Abandon();//结束 Session
    Response.Redirect("testGlobal.aspx");
}
```

(7) 编译并运行程序，在浏览器中打开 testGlobal.aspx 文件，效果如图 2-27 所示。从运行结果可以看出事件处理程序的执行顺序。

(8) 单击“刷新”按钮，由于此时 Session 尚未结束，所以页面内容显示的是“同一个 Session 时间”，如图 2-28 所示。

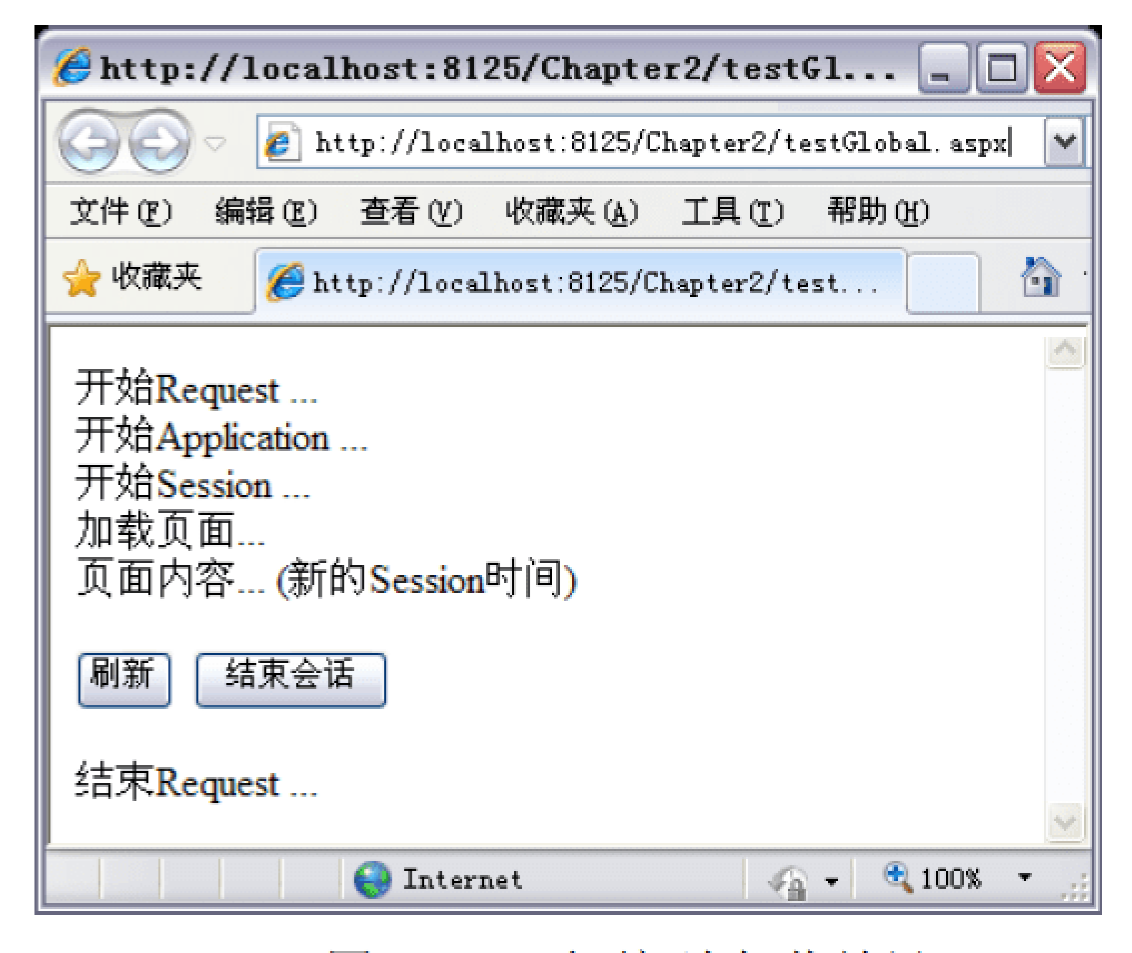


图 2-27　页面初次加载效果

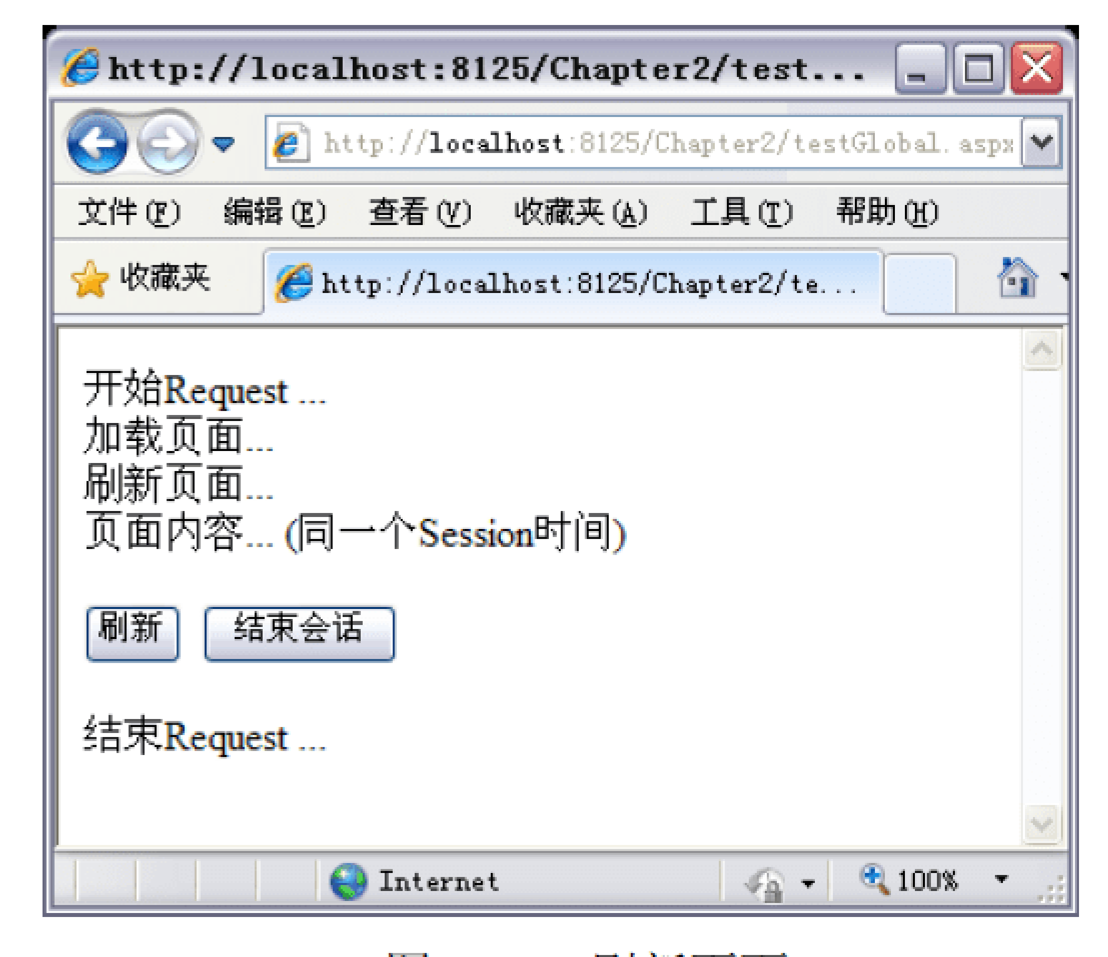


图 2-28　刷新页面

(9) 单击“结束会话”按钮，因为在程序中以 Abandon()方法强制结束 Session 事件，然后重新加载页面，可以看到，此次显示的为新的 Session 时间，如图 2-29 所示。

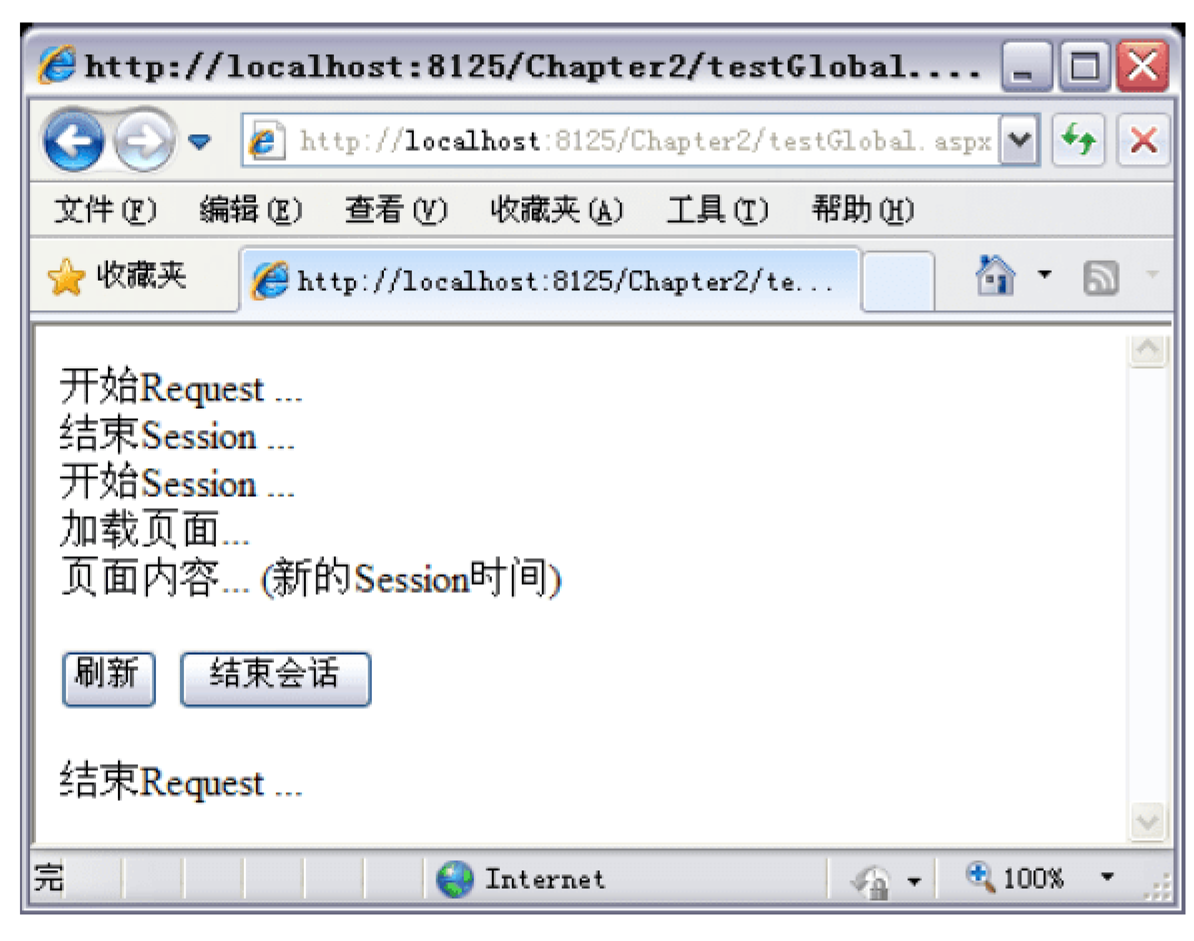


图 2-29　结束会话重新加载页面

2.4　本章小结

本章全面讲述了 ASP.NET 的基础知识。首先，介绍了 ASP.NET 应用程序的文件类型及其目录结构。接下来，对 ASP.NET 的内置对象分别进行了详细讲解，这些对象在今后的开发和学习过程中会经常用到，理解并掌握内置对象的使用有助于我们开发出安全健壮的应用程序。最后，讨论了 ASP.NET 的配置管理，包括 web.config 文件和 Global.asax 文件，正确使用配置文件可以给我们的开发工作带来很多便利。掌握本章的内容是后续学习的基础，下一章将开始 Web 窗体的设计开发，重点介绍 ASP.NET 的服务器控件和用户控件的使用。

2.5　思考和练习

1. App_Code 目录和 bin 目录有什么区别？
2. 简述加载页面时，Page 类各事件的发生顺序。
3. Response.Redirect()和 Server.Transfer()方法都能实现页面的重定向，二者有何区别？
4. 对于 HTML 表单，使用 Get 方法或 Post 方法提交数据有什么区别？
5. 在例 2-2 中的 Response.aspx 页面中，输出的 QueryString 集合字符串中出现了很多奇特的字符，这是为什么？如何让它正确显示提交的中文数据？
6. 新建一个网站，通过 Application 对象统计网站的访问人数。
7. web.config 文件是什么格式的？该配置文件包含哪些配置的设置。
8. Global.asax 文件的作用是什么？

第3章　ASP.NET服务器控件

ASP.NET 服务器控件是 ASP.NET 网页上的对象，使用 ASP.NET 服务器控件，可以大幅减少开发 Web 应用程序所需编写的代码量，提高开发效率和 Web 应用程序的性能。ASP.NET 服务器控件的体系结构已经完全集成到了 ASP.NET 中，为用户提供了一个在当今构建 Web 站点的技术中相当独特的功能集。本章将介绍这些服务器控件的基本用法以及不同类别控件的功能。这些控件在每个 ASP.NET 应用程序中都会用到。因此，了解工具箱中有哪些控件可用、它们各自的用途、它们的工作原理以及它们如何维持自身状态非常关键。

本章学习目标：

- ASP.NET 服务器控件的概念和工作原理
- 服务器控件的基类和常用事件
- 列表控件的使用
- 各种验证控件的功能和用法
- 使用 ASP.NET 的导航控件
- 如何创建和使用用户控件
- 为用户控件添加属性

3.1　服务器控件概述

在网页上经常看到填写信息用的文本框、单选按钮、复选框和下拉列表等元素，它们都是控件。控件是可重用的组件或对象，有自己的属性和方法，可以响应事件。

ASP.NET 服务器控件是服务器端 ASP.NET 网页上的对象，当用户通过浏览器请求 ASP.NET 网页时，这些控件将在服务器上运行，并向客户端呈现 HTML 标记。

在 ASP.NET 页面上，服务器控件表现为一个标记，例如<asp:textbox…/>。这些标记不是标准的 HTML 元素，因此如果它们出现在网页上，浏览器将无法理解它们，然而，当从 Web 服务器上请求一个 ASP.NET 页面时，这些标记都将动态地转换为 HTML 元素。

3.1.1　ASP.NET 页面的工作流程

ASP.NET 服务器控件是 ASP.NET 的重要组成部分。在 VWD 中构建的几乎所有页面都包含一个或多个服务器控件。这些控件有各种各样的类型和大小，有 Button 和 Label 这样的简单控件，也有复杂的控件，如可以显示数据源中数据的控件 TreeView 和 GridView。

在创建.aspx 页面时，可以将任意的服务器控件添加到页面中，然而客户端浏览器只能理解 HTML 和 JavaScript 脚本代码，因此，当从服务器上请求一个.aspx 页面时，服务器控件都会被动态地编译转换为标准的 HTML 元素。其工作流程如图 3-1 所示。

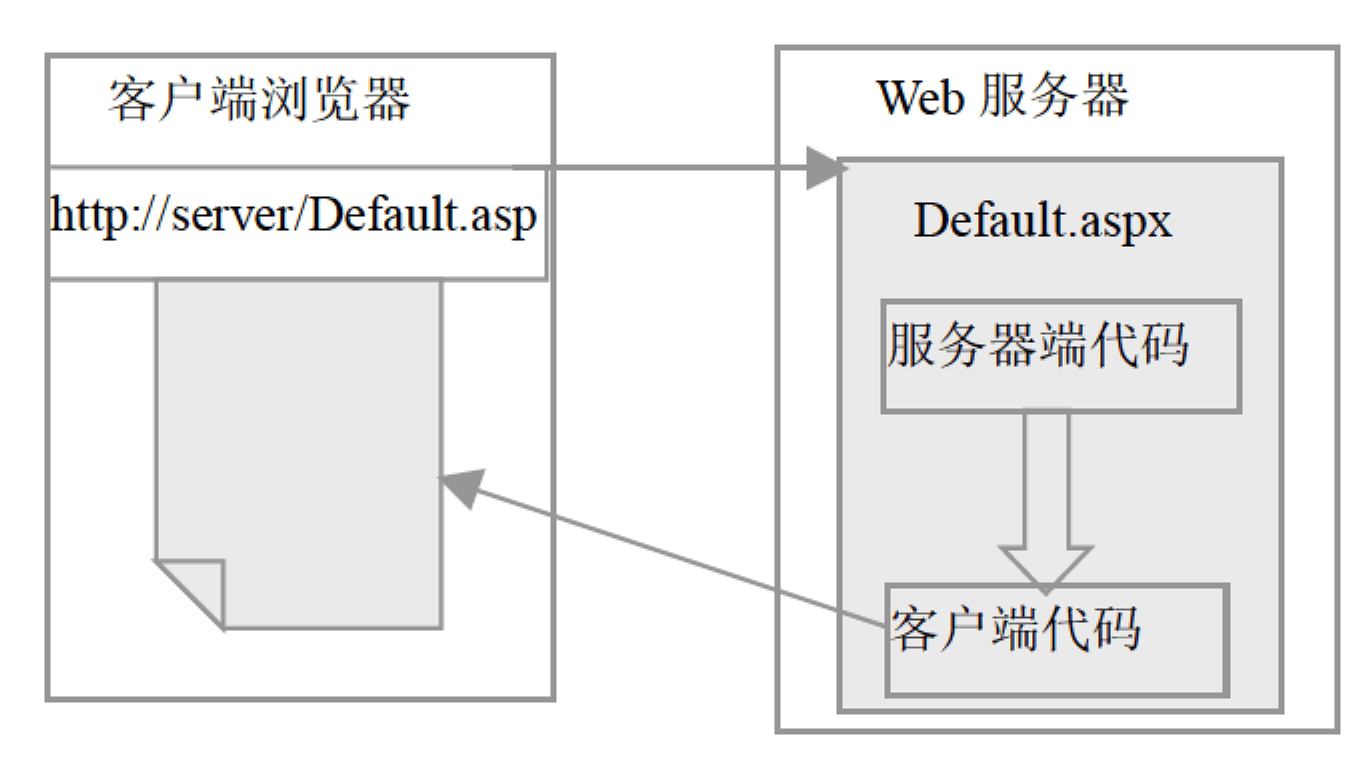


图 3-1　ASP.NET 页面的工作流程

3.1.2　服务器控件类

大多数 Web 服务器控件类都派生于 System.Web.UI.WebControls.WebControl 类，而 WebControl 类又从 System.Web.UI.Control 类派生而来。

WebControl 类定义了所有控件类的一些共同属性和事件。

1. 基本属性

WebControl 类是用作定义 System.Web.UI.WebControls 命名空间中的所有控件的公共方法、属性和事件的基类。其中定义了一些可以应用于几乎所有服务器控件的基本属性，如表 3-1 所示。

表 3-1　WebControl 类的基本属性

属　性	描　述
AccessKey	允许设置一个键，使用这个键，就可以按下关联的字母在客户端访问控件
BackColor	获取或设置 Web 服务器控件的背景色
BorderColor	获取或设置 Web 服务器控件的边框颜色
BorderStyle	获取或设置 Web 服务器控件的边框样式
BorderWidth	获取或设置 Web 服务器控件的边框宽度
CssClass	获取或设置 Web 服务器控件在客户端呈现的级联样式表(CSS)类
Enabled	获取或设置是否启用 Web 服务器控件，默认值为 True
EnableTheming	获取或设置是否对 Web 服务器控件应用主题
Font	获取或设置 Web 服务器控件关联的字体属性
ForeColor	获取或设置 Web 服务器控件的前景色，通常为文本颜色
Height	获取或设置 Web 服务器控件的高度

(续表)

属　　性	描　　述
ID	获取或设置 Web 服务器控件的编号标识符
SkinID	获取或设置应用于 Web 服务器控件的外观
Style	获取或设置将在 Web 服务器控件的外部标记上呈现的样式属性的文本属性的集合
TabIndex	设置客户端 HTML tabindex 特性，确定用户按下 Tab 键时焦点沿着页面中控件移动的顺序
ToolTip	允许设置浏览器中控件的工具提示。这个工具提示在 HTML 中被呈现为 title 特性，当用户把鼠标悬停在相关的 HTML 元素上时就会显示出来
Visible	获取或设置 Web 服务器控件是否作为 UI 呈现在页面上
Runat	该属性设置为 Server 时，表示该控件是一个服务器控件
Width	获取或设置 Web 服务器控件的宽度

在页面上添加了服务器控件之后，可以在“属性”窗口中修改控件的属性，也可以在“源”视图中直接设置不同的属性。

技巧：

“属性”窗口在“源”视图中也是可用的，只要简单地单击某标记，“属性”窗口就会更新，以反映输入的标记。通过“属性”窗口设置的属性，在“源”视图中也会自动生成相应的代码。

2. 服务器控件的事件

在 ASP.NET 页面中，用户与服务器的交互是通过 Web 控件的事件来完成的。例如，当单击一个按钮时，就会触发按钮的单击事件，程序员只需在该单击事件处理程序中编写相应的代码，即可对用户的单击行为做出响应。

服务器控件的事件工作方式与传统的 HTML 标记的客户端事件工作方式有所不同，这是因为 HTML 标记的客户端事件是在客户端触发并处理的，而 ASP.NET 中的 Web 控件的事件虽然也是在客户端触发，但却是在服务器端处理的。

Web 控件的事件模型，即客户端捕捉到事件信息，接着通过 HTTP POST 将事件信息发送到服务器，而且页面框架必须解释该 POST 以确定所发生的事件，然后在要处理该事件的服务器上调用代码中的相应方法。

基于以上事件模型，Web 控件事件可能会影响到页面的性能，因此，Web 控件仅提供有限的一组事件，常见的事件如表 3-2 所示。

表 3-2　服务器控件的事件

事　　件	支持的控件	描　　述
Click	Button、ImageButton	单击事件
TextChanged	TextBox	输入焦点变化
SelectedIndexChanged	DropDownList、ListBox、CheckBoxList、RadioButtonList	选择项变化

服务器控件通常不再支持经常发生的事件，如 OnMouseover 事件等，因为如果在服务器端处理这些事件，就会浪费大量的资源，但 Web 控件仍然可以为这些事件调用客户端处理程序。此外，控件和页面本身在每个处理步骤都会触发生命周期事件，如 Init 事件、Load 事件和 PreRender 事件，在应用程序中可以使用这些生命周期事件。

所有的 Web 事件处理函数都包括两个参数，第 1 个参数表示触发该事件的对象，第 2 个参数表示包含该事件特定信息的事件对象，通常是 EventArgs 类型，或 EventArgs 类型的子类型。例如，按钮控件的单击事件处理程序，其代码形式如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    //在此添加处理程序
}
```

3. 事件的绑定

在处理 Web 控件时，经常需要把事件绑定到事件处理程序。将事件绑定到事件处理程序的方法有如下两种。

(1) 在 ASP.NET 页面中，在声明控件时，指定该控件的事件对应的事件处理程序。例如，把一个 Button 控件的 Click 事件绑定到名为 MyClick 的方法，代码如下：

```
<asp:Button ID="Button1"  runat="server" Text="Button" onclick="MyClick" />
```

(2) 如果控件是动态创建的，则需要通过编写代码动态地将事件绑定到方法，例如：

```
Button myBtn = new Button("Button1");
myBtn.Text = "提交";
myBtn.Click += new System.EventHandler(ButtonClick);
```

在第 2 章本书曾介绍过在 VWD 中双击控件即可添加控件的默认事件的处理程序，也可以在“属性”窗口的“事件”选项卡页中添加具体的事件处理程序。这种操作其实是上面提到的第一种方法。

3.1.3　设置控件的颜色与字体

所有控件都有设置颜色和字体的属性，本节我们就来介绍如何设置颜色和字体。在.NET 框架中，System.Drawing 命名空间提供了一个 Color 类，使用它可以设置控件的颜色属性。

创建颜色的方式有如下 3 种。

- 使用 ARGB(alpha,red,green,blue)颜色值：可以为每个值指定一个 0~255 之间的整数。其中，alpha 表示颜色的透明度，当 alpha 为 255 时，表示完全不透明；red 表示红色；green 表示绿色；blue 表示蓝色。
- 使用颜色的枚举值：可供使用的颜色有 140 个。
- 使用 HTML 颜色名：可以使用 ColorTranslator 类把字符串转换为颜色值。

例如，下面的代码都是设置控件 Button1 的背景色：

```
Button1.BackColor = Color.FromArgb(255,0,99,127);
Button1.BackColor = Color.Red;
Button1.BackColor = ColorTranslator.FromHtml("Green");
```

控件的字体属性依赖于 System.Web.UI.WebControls 命名空间中的 FontInfo 对象。该对象的常用属性如表 3-3 所示。

表 3-3　FontInfo 对象的属性

属　性	描　述
Name	指明字体的名称，如 Arial
Names	指明一系列字体，浏览器会首先选用第一个去匹配用户安装的字体
Size	字体的大小，可以设置为相对值或者真实值
Bold、Italic、Strikeout、Underline、Overline	布尔属性，用来设置是否应用给定的样式特征。Bold 是粗体,Italic 为斜体，Strikeout 为中划线，Underline 为下划线，Overline 为上划线

技巧：
通常，可以直接在“属性”窗口中设置字体的上述属性。

3.1.4　服务器控件的类别

ASP.NET 4.0 本身附带了大量的服务器控件，能够满足 Web 开发的大部分需要。为了更容易地找到正确的控件，可以将它们放在工具箱的各个单独的控件类别中。如图 3-2 所示为工具箱中的所有可用类别。

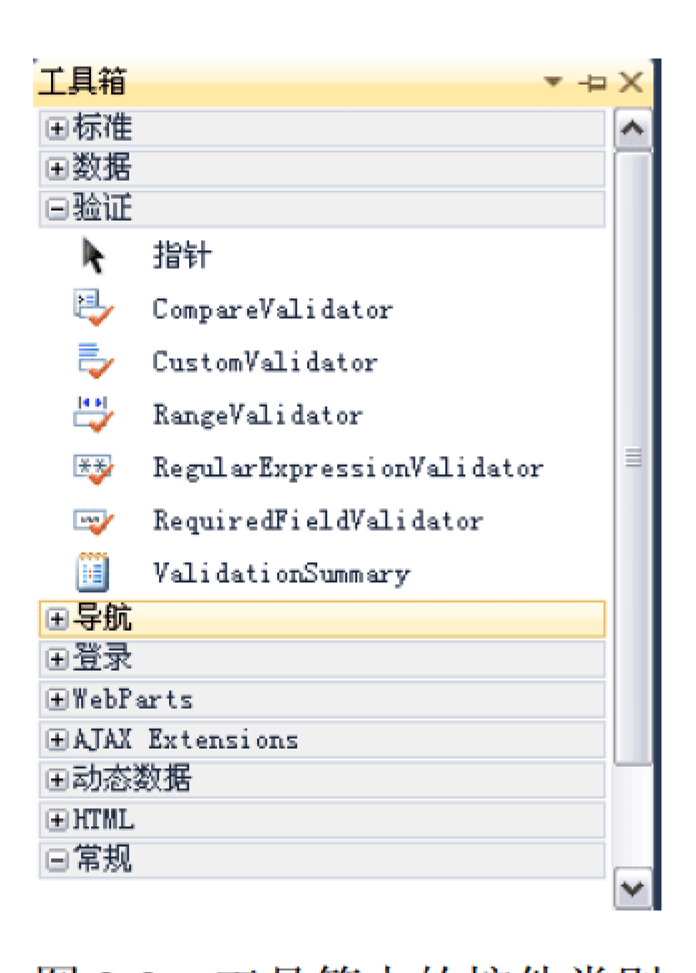


图 3-2　工具箱中的控件类别

本章将介绍标准控件、HTML 控件、验证控件和导航控件，数据控件将在第 5 章介绍，登录控件将在第 7 章介绍，AJAX Extensions 控件将在 8 章介绍，其他控件的使用本书将不做详细介绍。

3.2　标准控件

标准类别中包含很多基本控件，几乎所有的 Web 页面都需要它们。前面已经使用过其中的一部分，如 TextBox、Button 和 Label 控件。本节将详细介绍这些控件的功能和用法。

3.2.1　简单控件

这里所说的简单控件是指简单易懂且常用的控件，包括 Label、TextBox、Button、HyperLink、LinkButton、ImageButton、RadioButton 和 Checkbox。

1. Label 控件

Label 控件用来显示静态文本，其常用的属性主要有 ID、Text 和 Visible。其中 ID 和 Visible 属性是基础服务器控件基类的，Text 属性用于指定 Labe 控件要显示的文本。

2. TextBox 控件

TextBox 控件显示为一个输入文本框。默认情况下，该控件的 TextMode 属性设置为 TextBoxMode.SingleLine，即一个单行文本框。但也可以将 TextMode 属性设置为 TextBoxMode.MultiLine(显示多行文本框，该文本框将作为 textarea 元素呈现)或者为 TextBoxMode.Password(显示屏蔽用户输入的文本框，即密码框)。通过使用 Text 属性可以获得 TextBox 控件中的文本。

说明：

将 TextMode 属性设置为 TextBoxMode.Password 有助于确保在输入密码时其他人无法看到。但是，输入到文本框中的文本没有以任何方式进行加密，为了提高安全性，在发送带有密码的页面时，可以使用安全套接字层(SSL)和加密。

3. 按钮控件

按钮控件为用户提供向服务器发送命令的功能，它将窗体提交给服务器并使窗体同任何挂起的事件一起被处理。

ASP.NET 包括 3 种按钮控件：标准按钮、超级链接按钮和图形化按钮。这 3 种按钮提供类似的功能，但具有不同的外观。

- Button：显示一个标准命令按钮，该按钮呈现为一个 HTML input 元素。
- LinkButton：呈现为页面中的一个超链接。但是，它包含使窗体被发回服务器的客户端脚本(可以使用 HyperLink 服务器控件创建真实的超链接)。
- ImageButton：将一个图形呈现为按钮。这对于提供丰富的按钮外观非常有用。ImageButton 控件还提供有关图形内已单击位置的坐标信息，主要的属性是 ImageUrl。

当用户单击任何按钮服务器控件时，都会将该页发送到服务器。默认情况下，该页回发到其本身，在这里重新生成相同的页面并处理该页上控件的事件处理程序。可以引发按钮的 Click 事件，为这些事件编写事件处理程序。

可以配置按钮来将当前页面回发到另一页面。这对于创建多页窗体可能非常有用。

注意：

默认情况下，Button 控件使用 HTML POST 操作提交页面。LinkButton 和 ImageButton 控件不能直接支持 HTML POST 操作。因此，使用这些按钮时，它们将客户端脚本添加到页面以允许控件以编程方式提交页面(因此 LinkButton 和 ImageButton 控件要求在浏览器上启用客户端脚本)。

在某些情况下，可能希望 Button 控件也使用客户端脚本执行回发。这在希望以编程方式操作回发(如将回发附加到页面上的其他元素)时非常有用。可以将 Button 控件的 UseSubmitBehavior 属性设置为 true 以使 Button 控件使用基于客户端脚本的回发。

Button 控件既可以引发服务器事件，也可以引发客户端事件。服务器事件在回发后发生，且这些事件在为页面编写的服务器端代码中处理。客户端事件在客户端脚本(通常为 ECMAScript(JavaScript))中处理，并在提交页面前引发。通过向 ASP.NET 按钮控件添加客户端事件，可以执行一些任务(如在提交页之前显示确认对话框以及可能取消提交)。

4. HyperLink 控件

HyperLink 控件用来创建一个超链接，该控件在客户端将呈现为一个 HTML<a>元素，其常用的属性主要有 ID、Text、Target 和 NavigateUrl。其中 Text 属性用于指定超链接显示的文本，Target 属性用于设置单击超链接时显示网页的目标窗口或框架，NavigateUrl 属性用于指定链接的 URL 地址。

5. RadioButton 和 Checkbox

RadioButton 控件用于显示一个单选按钮；CheckBox 控件用于显示一个复选框。这两个控件都对应有列表控件 RadioButtonList 和 CheckBoxList，当选项较多或需要在运行时动态决定有哪些选项时，使用列表控件 RadioButtonList 和 CheckBoxList 控件比较方便。

单选按钮很少单独使用，而是进行分组以提供一组互斥的选项。在一个组内，每次只能选择一个单选按钮，可以将所有 RadioButton 控件的 GroupName 属性设置为相同的组名即可将单选按钮进行分组。

当用户单击 RadioButton 或 Checkbox 控件时将引发 CheckedChanged 事件。默认情况下，这一事件并不导致向服务器发送页面，但通过将 AutoPostBack 属性设置为 true，可以使该事件强制立即发送。

6. 使用简单控件

在前面的学习中，已经使用过 Label、Button 和 TextBox 控件了，要向页面中添加服务器控件，只需简单地从工具箱中拖动相应的控件到设计视图中即可。

下面通过一个具体的实例来看一下简单服务器控件的工作原理。

例 3-1：使用简单服务器控件。在本例中，将向页面添加一些简单控件。当客户端通过浏览器请求该页面时，会将这些服务器控件转换为 HTML，然后再发送给客户端。如果在浏览器中查看页面的源文件，就能发现这些 HTML 完全不同于初始的 ASP.NET 标记。

(1) 启动 VWD2010，选择“文件”|“新建网站”命令，新建一个空网站 Chapter3，单击“确定”按钮。

(2) 选择“网站” | “添加新项”命令，打开“添加新项”对话框，添加一个名为 Default.aspx 的 Web 窗体。

(3) 切换到页面的“设计”视图，从“工具箱”中拖动本节介绍的简单控件到 Web 窗体中，控件的 Text 属性和布局如图 3-4 所示。

(4) TextBox2 在此是作为密码输入框的，所以需要设置其 TextMode 属性为 Password，如图 3-4 所示。

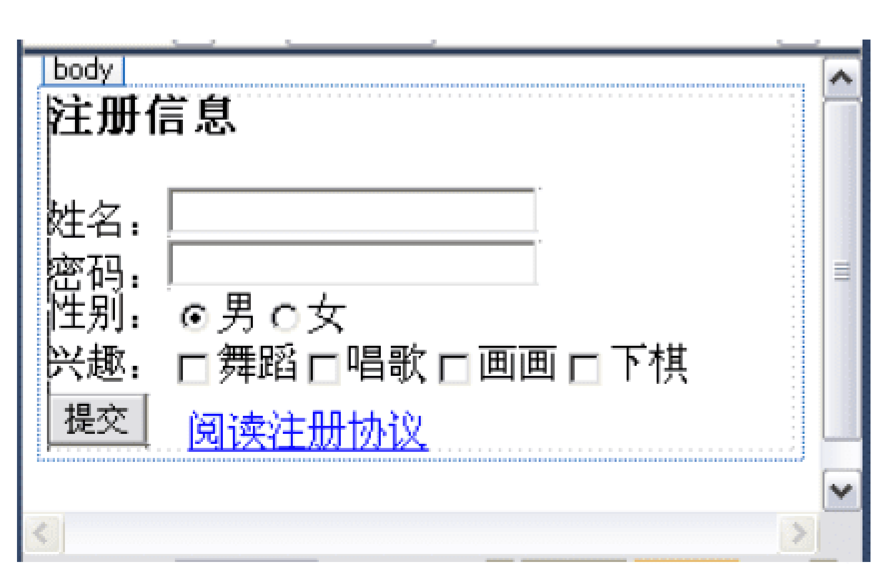


图 3-3　窗体中的控件布局

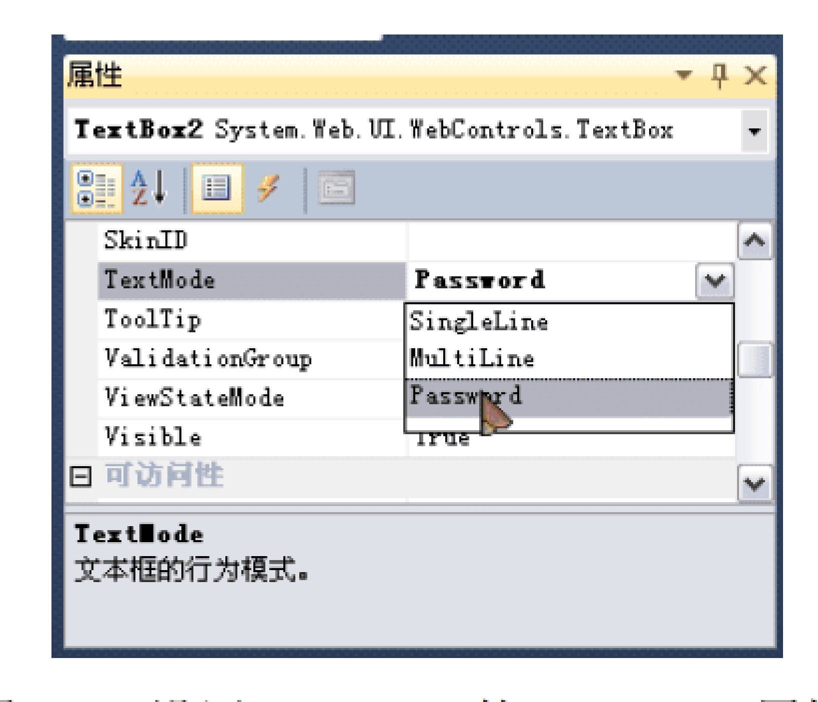


图 3-4　设置 TextBox2 的 TextMode 属性

(5) 设置两个 RadioButton 控件的 GroupName 均为 gender，这样，两个单项按钮在同一时刻只能有一个被选中。

(6) 设置 HyperLink 控件的 NavigateUrl 为一个超链接地址，用于打开注册协议。

(7) 双击“提交”按钮，添加按钮的单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text == "" || TextBox2.Text == "")
    {
        string info = "alert(\"请输入姓名和密码！ \"); ";
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", info, true);
    }
    else
    {
        string str = "";
        if(CheckBox1.Checked)
            str += "舞蹈　";
        if(CheckBox2.Checked)
            str += "唱歌　";
        if(CheckBox3.Checked)
```

```
                str += "画画　";
            if(CheckBox4.Checked)
                str += "下棋　";

Response.Redirect(string.Format("success.aspx?user={0}&password={1}&gender={2}
                &favor={3}",TextBox1.Text, TextBox2.Text,RadioButton1.Checked?"男":"女
",str));
        }
    }
```

这段代码中，首先检查 TextBox 1 和 TextBox 2 中的内容是否为空，若为空，则通过 alert 语句弹出对话框，提示用户输入姓名和密码；若不为空，则将控件中的数据作为参数传给 success.aspx 页面。所以我们还必须添加 success.aspx 页面。

(8) 通过“添加新项”对话框，添加一个名为 success.aspx 的 Web 窗体。

(9) 在 success.aspx 页面的 Load 事件中添加如下代码：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("注册成功！<br/>注册信息如下：<br/>姓名：");
        Response.Write(Request.QueryString["user"].ToString());
        Response.Write("<br/>密码：");
        Response.Write(Request.QueryString["password"].ToString());
        Response.Write("<br/>性别：");
        Response.Write(Request.QueryString["gender"].ToString());
        Response.Write("<br/>兴趣：");
        Response.Write(Request.QueryString["favor"].ToString());
    }
```

(10) 编译并运行程序，在浏览器中加载页面 Default.aspx，如图 3-5 所示。如果不输入姓名和密码，就单击“提交”按钮，将弹出提示对话框，如图 3-6 所示。

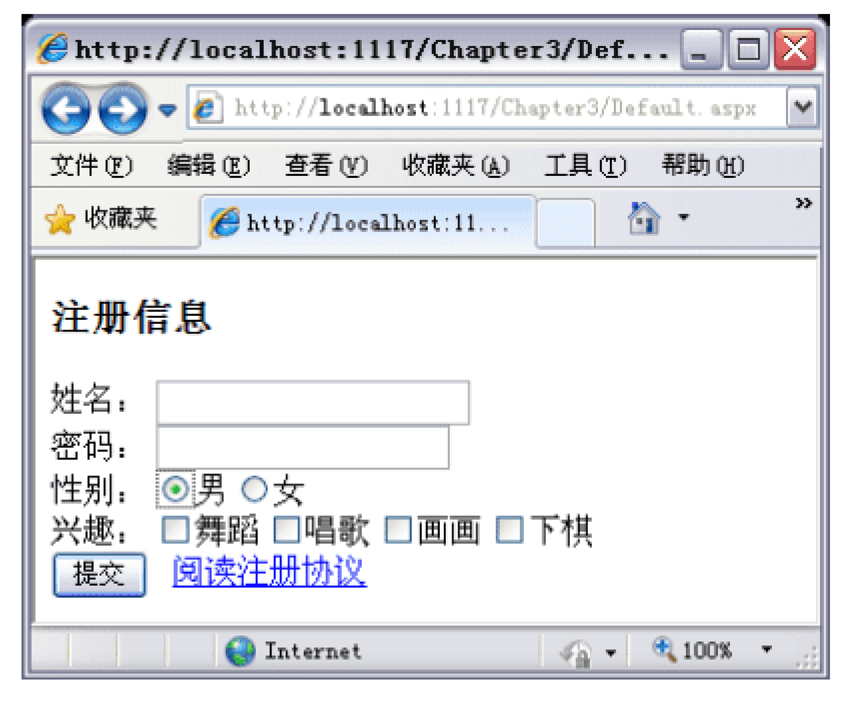




图 3-5　页面效果

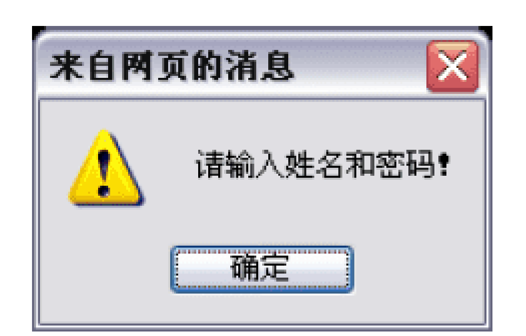


图 3-6　提示对话框功

(11) 输入注册信息后，单击“提交”按钮，将跳转到注册成功页面 success.aspx，显示注册信息，如图 3-7 所示。

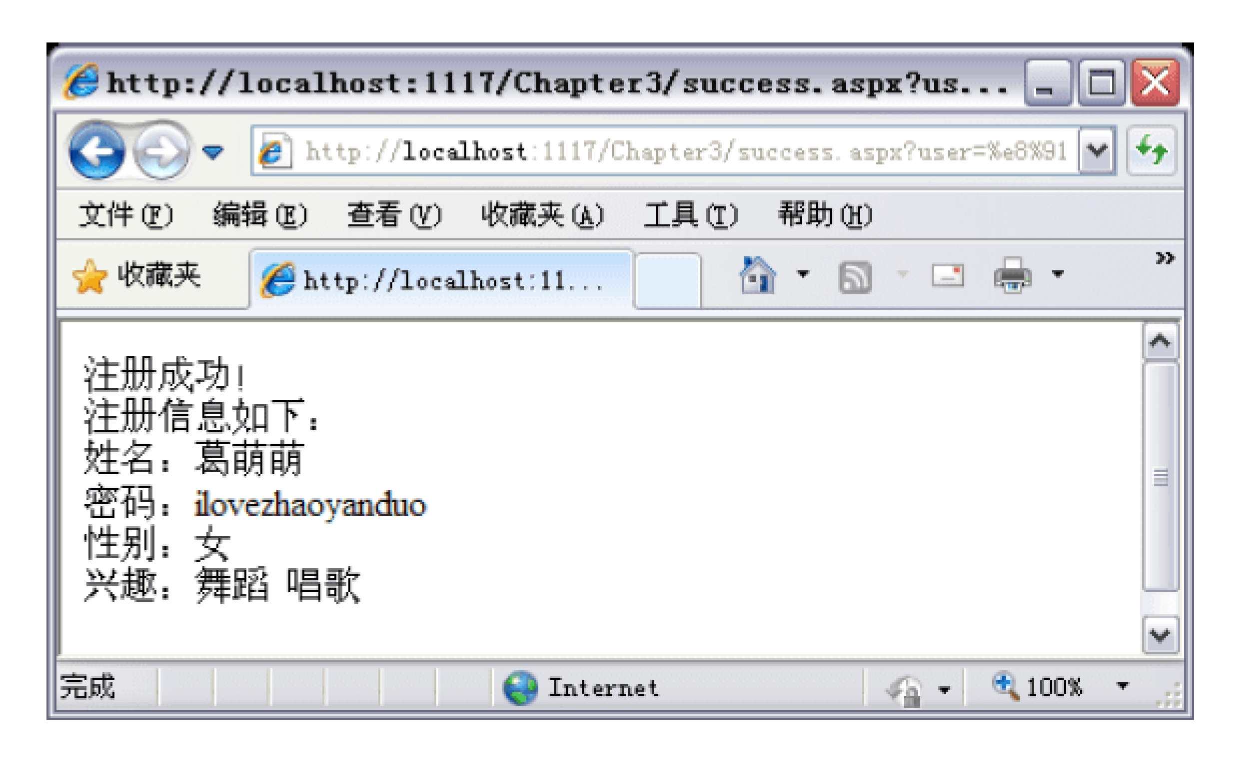


图 3-7　注册成功

下面来看一下服务器控件的工作原理。当在浏览器中请求页面时，服务器端控件就由 ASP.NET 运行库(负责接收和处理 ASPX 页面请求的引擎)处理。然后控件就会输出客户端 HTML 代码，并将其附加到最终页面输出的后面。最终出现在浏览器中用来构建页面的就是该 HTML 代码。例如，当首次加载 Label 控件并请求它的 HTML 时，它会返回下面的代码：

```
<span id="Label1">姓名</span>
```

从上面这行代码可以看出，虽然使用<asp:Label>语法定义了 Label 控件，但是它最终出现在浏览器中的只是一个简单的<span>元素。在<span>标记中显示的是 Label 控件的 Text 属性值。

当请求例 3-1 的 Default.aspx 页面后，可以通过“查看”|“源文件”命令查看每个控件生成的 HTML 代码，如下所示：

```
<div>
    <h3>注册信息</h3>
    <span id="Label1">姓名：</span>
    <input name="TextBox1" type="text" id="TextBox1" />
    <br />
    <span id="Label2">密码：</span>
    <input name="TextBox2" type="password" id="TextBox2" />
    <br />
    <span id="Label3">性别：</span>
    <input id="RadioButton1" type="radio" name="gender" value="RadioButton1"
checked="checked" /><label for="RadioButton1">男</label>
    <input id="RadioButton2" type="radio" name="gender" value="RadioButton2" /><label
for="RadioButton2">女</label>
    <br />
    <span id="Label4">兴趣：</span>
    <input id="CheckBox1" type="checkbox" name="CheckBox1" /><label
for="CheckBox1">舞蹈</label>
    <input id="CheckBox2" type="checkbox" name="CheckBox2" /><label
for="CheckBox2">唱歌</label>
```

```
                <input id="CheckBox3" type="checkbox" name="CheckBox3" /><label
for="CheckBox3">画画</label>
                <input id="CheckBox4" type="checkbox" name="CheckBox4" /><label
for="CheckBox4">下棋</label>
                <br />
                <input type="submit" name="Button1" value="提交" id="Button1" />
                 
                <a id="HyperLink1" href="xieyi.htm">阅读注册协议</a>
                <br />
            </div>
```

3.2.2　列表控件

标准类别中有许多在浏览器中表现为列表的控件。这些控件包括 ListBox、DropDownList、CheckBoxList、RadioButtonList 和 BulletedList。要向列表中添加项，可以在控件的起始和结束标记之间定义<asp:ListItem>元素，如下面的示例所示：

```
<asp:DropDownList ID="FavoriteLanguage" runat="server">
  <asp:ListItem Value="C#">C#</asp:ListItem>
  <asp:ListItem Value="Visual Basic">Visual Basic</asp:ListItem>
  <asp:ListItem Value="CSS">CSS</asp:ListItem>
</asp:DropDownList>
```

DropDownList、RadioButtonList 控件允许用户一次只能选择一项。要以编程方式查看列表控件中当前活动和选中的项，可以查看它的 SelectedValue、SelectedItem 或 SelectedIndex 属性。SelectedValue 返回一个包含选中项的值的字符串，SelectedIndex 返回列表中项基于 0 的索引。

RadioButtonList 控件不允许在按钮之间插入文本，但如果想将按钮绑定到数据源，使用这类控件将非常方便。

对于允许多重选择的控件，CheckBoxList 和 ListBox，可以在 Items 集合之间循环，并且查看选中了哪些项。在这种情况下，SelectedItem 和 SelectedValue 仅返回列表中第一个选中的项；而不是返回所有选中项。

BulletedList 控件不允许用户作选择，因而不支持 SelectedValue、SelectedItem 或 SelectedIndex 这些属性。

当列表控件的某个选项被选中时，该控件将引发 SelectedIndexChanged 事件。默认情况下，此事件不会导致向服务器发送页，但可以通过将 AutoPostBack 属性设置为 true，强制该控件立即发送。

例 3-2：演示列表控件的使用。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框在该网站中添加一个名为 List.aspx 的 Web 窗体页。

(2) 打开 List.aspx 文件的“设计”视图，从“工具箱”中拖动 DropDownList、RadioButtonList、CheckBoxList、ListBox、BulletedList 控件各一个到 Web 窗体中，控件的 Text 属性和布局如图 3-8 所示。

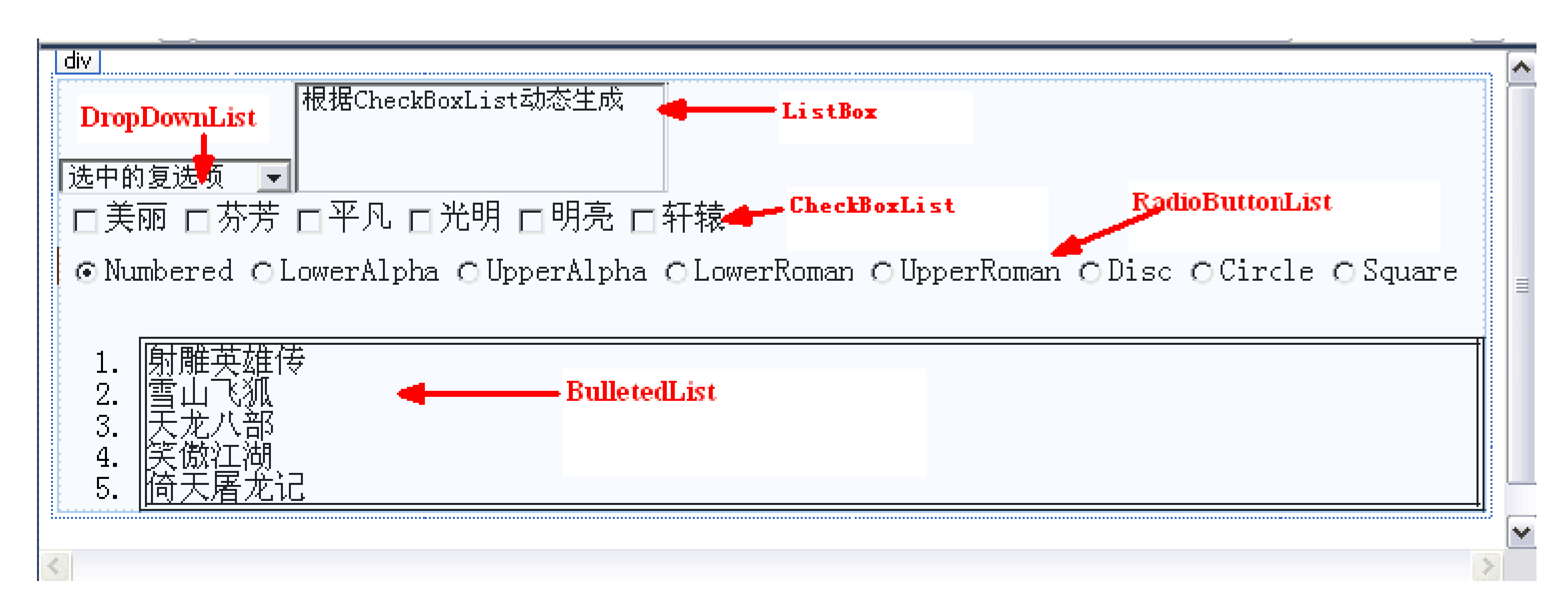


图 3-8　窗体中的控件布局

(3) 设置列表控件的 Items 属性时，可单击“属性”窗口中 Items 属性右边的...按钮，如图 3-9 所示。此时将弹出“ListItem 集合编辑器”对话框，在该对话框中，单击“添加”按钮，然后在 Text 和 Value 文本框中输入相应的列表项即可，如图 3-10 所示。通过此窗口添加的项将被添加为控件的标记之间的<asp:ListItem>元素。






图 3-9　设置列表控件的 Items 属性　　　　图 3-10　“ListItem 集合编辑器”对话框

(4) 设置 DropDownList、CheckBoxList 和 RadioButtonList 控件的 AutoPostBack 属性。选中某个列表控件，单击右上角的箭头图标，打开控件的“任务”菜单，通过该菜单可以执行属于控件的大部分常见任务。如图 3-11 所示有 3 个选项：第一个选项允许把控件与数据源绑定在一起；第二个选项用来编辑列表项，与前面设置 Items 属性相同；最后一个选项用来设置控件的 AutoPostBack 属性。选中这个选项后，一旦用户从列表中选择了一个新项，控件就会将它包含的页面提交回服务器。

(5) 为 DropDownList、CheckBoxList 和 RadioButtonList 控件添加 SelectedIndexChanged 事件处理程序。首先选中控件，然后在“属性”窗口中单击工具按钮，切换到事件列表，如图 3-12 所示。在相应的事件后面的文本框中双击即可添加默认名称的事件处理程序。

说明：

本例中，DropDownList 和 CheckBoxList 控件的事件响应方法是同一个，即 DropDownList1_SelectedIndexChanged。在该处理方法中，将根据 DropDownList 控件的选项判断 ListBox 控件中所显示的选项，是 CheckBoxList 控件中的选中项还是非选中项。



图 3-11　列表控件的“任务”菜单

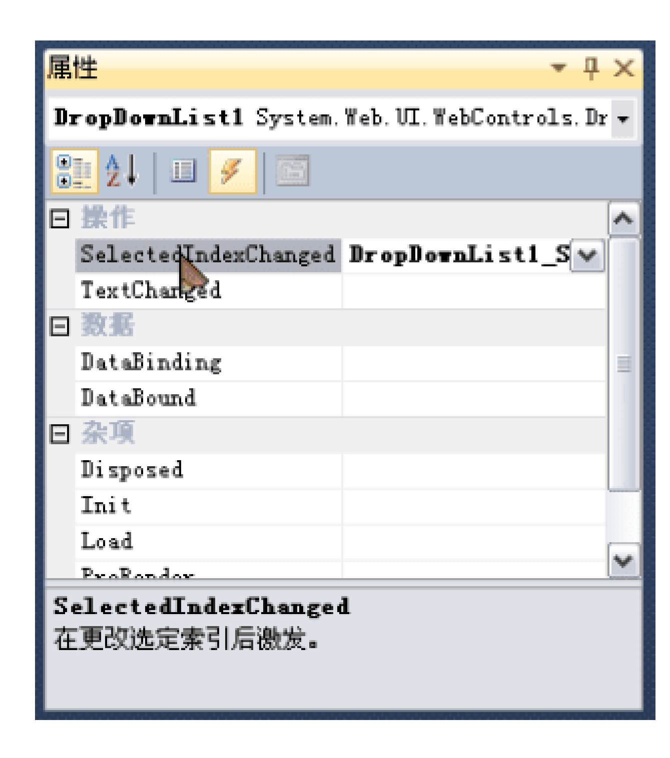


图 3-12　为控件添加事件处理程序

(6) 设置 CheckBoxList 和 RadioButtonList 控件的 RepeatDirection 属性为 Horizontal，使选项水平排列。

(7) 设置 ListBox 控件的 SelectionMode 属性为 Multiple，以允许用户进行多项选择。

(8) 设置完控件的属性和事件后，在 List.aspx 的“源”视图中可以看到如下代码：

```
<div>
    <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
        onselectedindexchanged="DropDownList1_SelectedIndexChanged">
        <asp:ListItem>选中的复选项</asp:ListItem>
        <asp:ListItem>非选中的复选项</asp:ListItem>
    </asp:DropDownList>
    <asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
        <asp:ListItem>根据 CheckBoxList 动态生成</asp:ListItem>
    </asp:ListBox>
    <asp:CheckBoxList ID="CheckBoxList1" runat="server"
        RepeatDirection="Horizontal"
        onselectedindexchanged="DropDownList1_SelectedIndexChanged"
        AutoPostBack="True">
        <asp:ListItem>美丽</asp:ListItem>
        <asp:ListItem>芬芳</asp:ListItem>
        <asp:ListItem>平凡</asp:ListItem>
        <asp:ListItem>光明</asp:ListItem>
        <asp:ListItem>明亮</asp:ListItem>
        <asp:ListItem>轩辕</asp:ListItem>
    </asp:CheckBoxList>
    <asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
        onselectedindexchanged="RadioButtonList1_SelectedIndexChanged"
        RepeatDirection="Horizontal">
        <asp:ListItem Selected="True">Numbered</asp:ListItem>
        <asp:ListItem>LowerAlpha</asp:ListItem>
        <asp:ListItem>UpperAlpha</asp:ListItem>
        <asp:ListItem>LowerRoman</asp:ListItem>
        <asp:ListItem>UpperRoman</asp:ListItem>
```

```
                <asp:ListItem>Disc</asp:ListItem>
                <asp:ListItem>Circle</asp:ListItem>
                <asp:ListItem>Square</asp:ListItem>
            </asp:RadioButtonList>
            <asp:BulletedList ID="BulletedList1" runat="server" BorderStyle="Double"
                BulletStyle="Numbered">
                <asp:ListItem>射雕英雄传</asp:ListItem>
                <asp:ListItem>雪山飞狐</asp:ListItem>
                <asp:ListItem>天龙八部</asp:ListItem>
                <asp:ListItem>笑傲江湖</asp:ListItem>
                <asp:ListItem>倚天屠龙记</asp:ListItem>
            </asp:BulletedList>
        </div>
```

(9) 在 List.aspx.cs 文件中，添加控件的事件处理程序，代码如下：

```
        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
        {
            ListBox1.Items.Clear();
            if (DropDownList1.SelectedValue == "选中的复选项")
            {
                foreach (ListItem item in CheckBoxList1.Items)
                {
                    if (item.Selected)
                        ListBox1.Items.Add(item);
                }
            }
            if (DropDownList1.SelectedValue == "非选中的复选项")
            {
                foreach (ListItem item in CheckBoxList1.Items)
                {
                    if (!item.Selected)
                        ListBox1.Items.Add(item);
                }
            }
        }
        protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
        {
            BulletStyle style = (BulletStyle)Enum.Parse(typeof(BulletStyle),
RadioButtonList1.SelectedValue);
            BulletedList1.BulletStyle = style;
        }
```

(10) 编译并运行程序，在浏览器中加载页面 List.aspx，如图 3-13 所示，读者可以选择不同的选项，查看控件的变化情况。

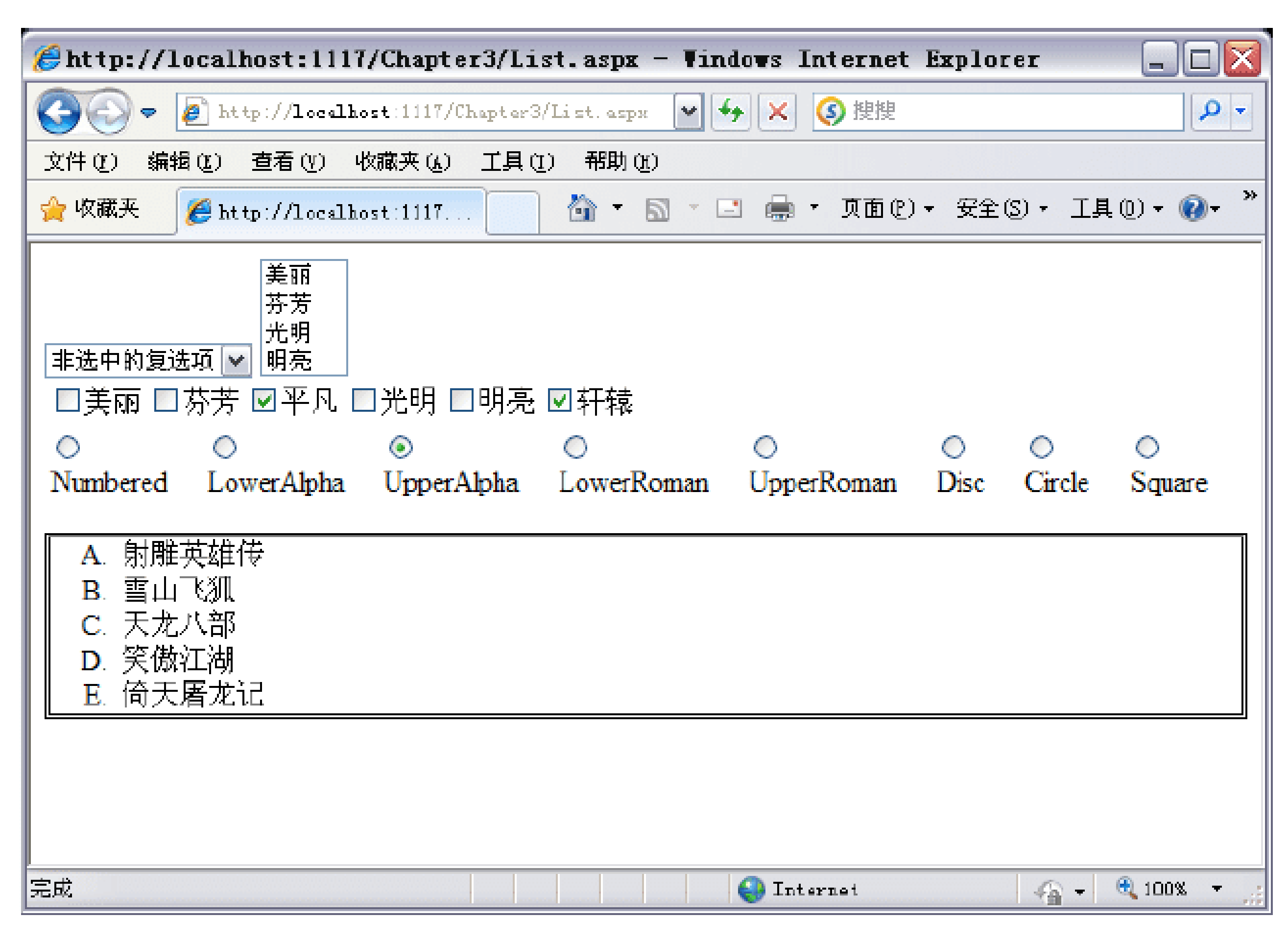


图 3-13　页面运行效果

3.2.3　容器控件

容器控件常用于以某种方式将相关的内容和控件组合到一起，常用的容器控件包括 Panel、PlaceHolder、MultiView、View 和 Wizard。例如，可以使用 PlaceHolder 或 Panel 控件同时隐藏或显示几个控件。不用分别隐藏每个控件，只需隐藏包含各个控件和标记的整个容器即可。这两个控件各有优缺点。PlaceHolder 控件的好处是它不会向页面发布它自己的 HTML，因此可以用作容器控件，而不会在最终页面中产生任何副作用。然而，它缺少设计时支持，因此在 VWD 中难以在设计时管理 PlaceHolder 内的控件。而 Panel 控件允许轻松地访问所有控件以及它所包含的其他内容，但是它自己则呈现为<div>标记，因此，一般常使用 Panel 控件。

使用 MultiView 和 View 控件可以制作出选项卡的效果，MultiView 控件用作一个或多个 View 控件的外部容器。View 控件又可以包含标记和控件的任何组合。

如果要切换视图，可以使用控件的 ID 或者 View 控件的索引值。在 MultiView 控件中，一次只能将一个 View 控件定义为活动视图。如果某个 View 控件定义为活动视图，那么它所包含的子控件则会呈现到客户端。可以使用 ActiveViewIndex 属性或 SetActiveView 方法定义活动视图。

注意：

如果 ActiveViewIndex 属性为空，则 MultiView 控件不向客户端呈现任何内容。如果活动视图设置为 MultiView 控件中不存在的 View，则会引发 ArgumentOutOfRangeException 异常。

无论是 MultiView 控件还是各个 View 控件，除当前 View 控件的内容外，都不会在页面中显示任何标记。但是，每次呈现页面时都会创建所有 View 控件中的所有服务器控件的实例，并且将这些实例的值存储为页面视图状态的一部分。另外，可以将一个主题分配给 MultiView 或 View 控件，控件将该主题应用于当前 View 控件的所有子控件。

MultiView 和 Wizard 相似的地方是：它们允许将一个长页面划分为多个区域。例如，将某个网上购物分成多个子步骤完成。区别在于 Wizard 具有使用 Previous、Next 和 Finish 按钮在页面间移动的内置支持，而 MultiView 则必须通过编程进行控制。

例 3-3：使用容器控件。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框在该网站中添加一个名为 Container.aspx 的 Web 窗体页。

(2) 打开 Container.aspx 文件的“设计”视图，从“工具箱”中拖动一个 CheckBox 控件和一个 Panel 控件到 Web 窗体中，然后在 Panel 中添加一个 Label 控件，切换到“源”视图，修改其 HTML 代码如下：

```
<asp:CheckBox ID="CheckBox1" runat="server"
    oncheckedchanged="CheckBox1_CheckedChanged" Text="显示 Panel 控件"
    AutoPostBack="True" />
<asp:Panel ID="Panel1" runat="server" Visible="False">
    <asp:Label ID="Label1" runat="server" Text="我是 Panel 控件中的 Label 控件
"></asp:Label>
</asp:Panel>
```

(3) 接着，输入文本信息“MultiView 与 View 控件演示…”，然后添加一个 RadioButtonList 控件和一个 MultiView 控件，在 RadioButtonList 控件中添加 3 个选项：View1、View2、View3。设置 RadioButtonList 控件的 AutoPostBack 属性为 True，RepeatDirection 属性为 Horizontal，并为其添加 SelectedIndexChanged 事件处理程序。

(4) 在 MultiView 控件中添加 3 个 View 控件，分别单击 3 个 View 控件，在每个 View 控件中分别输入不同的文本，以区分不同的 View 控件，如“这是 View1 控件”、“这是 View2 控件”和“这是 View3 控件”。切换到“源”视图，其 HTML 代码如下：

```
<h4>MultiView 与 View 控件演示...</h4>
<asp:RadioButtonList ID="RadioButtonList1" runat="server"
    RepeatDirection="Horizontal" AutoPostBack="True"
    onselectedindexchanged="RadioButtonList1_SelectedIndexChanged">
    <asp:ListItem Value="0">View1</asp:ListItem>
    <asp:ListItem Value="1">View2</asp:ListItem>
    <asp:ListItem Value="2">View3</asp:ListItem>
</asp:RadioButtonList>
<asp:MultiView ID="MultiView1" runat="server">
<asp:View ID="View1" runat="server">
    这是 View1 控件
</asp:View>
<asp:View ID="View2" runat="server">
    这是 View2 控件
</asp:View>
<asp:View ID="View3" runat="server">
    这是 View3 控件
```

```
        </asp:View>
        </asp:MultiView>
```

(5) 最后，添加一个 Wizard 控件。单击控件右上角的箭头打开“Wizard 任务”面板，选择“添加/删除 WizardSteps”命令，如图 3-14 所示。将打开“WizardStep 集合编辑器”对话框，如图 3-15 所示。

(6) 单击对话框左边“成员”列表中名为 Step1 的第一个 WizardStep，将它的 Title 属性修改为“选择银行”，将第二步的 Title 设置为“输入银行卡信息”，将第三步设置为“完成”。

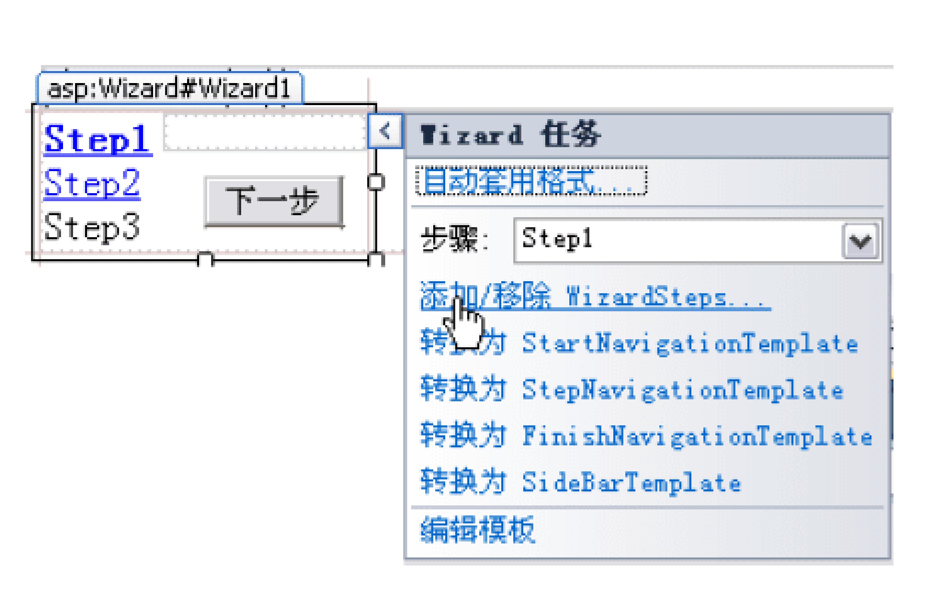


图 3-14　“Wizard 任务”面板



图 3-15　“WizardStep 集合编辑器”对话框

(7) 将第二步的 StepType 设置为 Finish，第三步的 StepType 设置为 Complete。可以让第一步的 StepType 保持为 Auto。单击“确定”按钮，关闭“WizardStep 集合编辑器”对话框。

(8) 在“设计”视图中，单击左边列表中的“选择银行”，让它成为活动步骤，然后添加一个 DropDownList 控件到 Wizard 控件中，并添加支持的银行信息。

注意：

向 Wizard 控件中添加其他控件时，需要将控件拖到 Wizard 右上角的灰色矩形框内，否则控件最后不会出现在 Wizard 内。

(9) 用同样的方法，在“输入银行卡信息”步骤中添加相应的控件，如图 3-16 所示，在“完成”步骤中添加一个 Label 控件，用于显示提示信息。

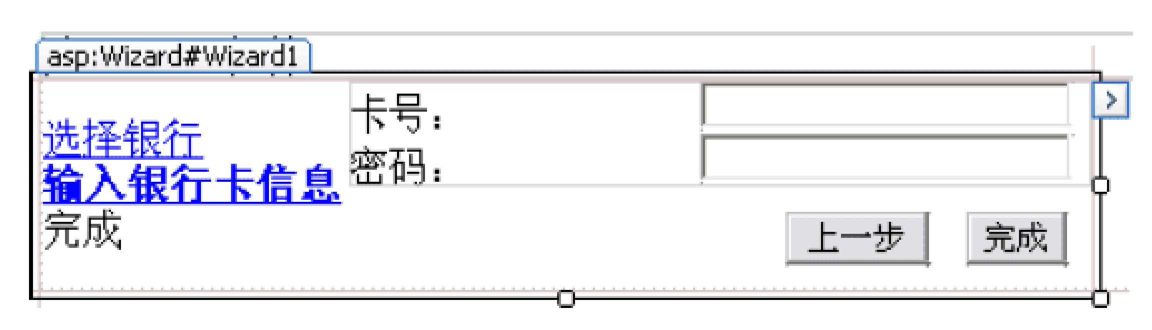


图 3-16　“输入银行卡信息”步骤的设计视图

(10) 当用户单击向导最后一步的“完成”按钮时需要进行相应的事件处理。打开控件的“属性”面板，选择“事件”选项卡。定位并双击 Action 类别中的 FinishButtonClick，为其添加事件处理程序。

```
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    Label2.Text = " 你好，支付信息如下：<br/>银行：";
    Label2.Text += DropDownList1.SelectedValue;
    Label2.Text += "<br/>卡号：";
    Label2.Text += TextBox1.Text + "<br/>密码：";
    Label2.Text += TextBox2.Text;
}
```

(11) 为复选框控件添加 CheckedChanged 事件处理程序，代码如下：

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    Panel1.Visible = CheckBox1.Checked;
}
```

(12) 为 RadioButtonList 控件添加 SelectedIndexChanged 事件处理程序，代码如下：

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
    MultiView1.ActiveViewIndex = Int32.Parse(RadioButtonList1.SelectedValue);
}
```

(13) 编译并运行程序，在浏览器中加载页面 Container.aspx，通过“显示 Panel 控件”复选框可以显示或隐藏 Panel 控件，通过 RadioButtonList 控件的选项可以显示不同的 View 控件。

(14) 在 Wizard 控件中的每个步骤中选择并输入相应的信息，然后单击“完成”按钮，在“完成”步骤中将显示前面两步中选择的支付信息，如图 3-17 所示。

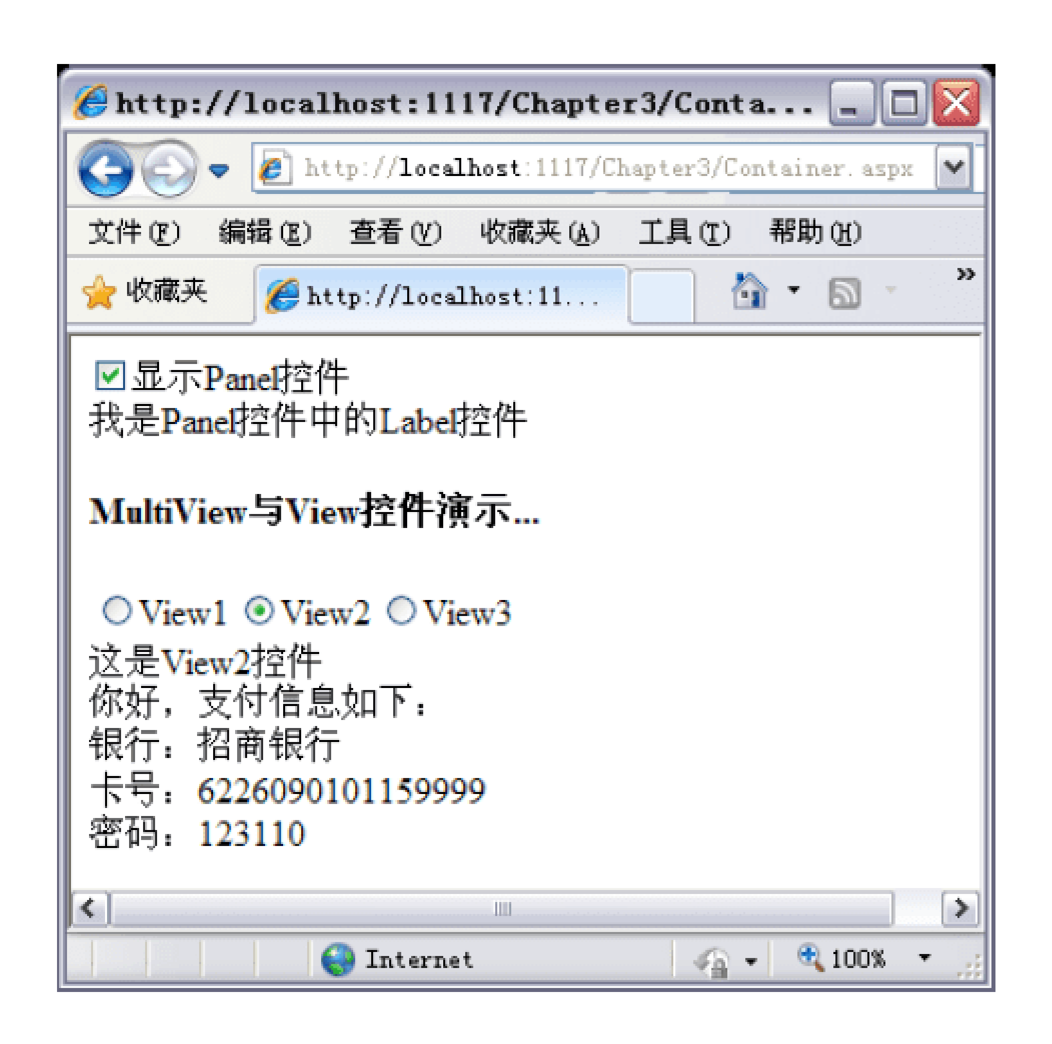


图 3-17　页面运行效果

3.2.4　其他标准控件

除了前面介绍的简单控件、列表控件和容器控件，标准控件中还有很多其他控件。它们的用法也都类似，在此只对这些控件做简单介绍。

1. Image 和 ImageMap

这两个控件用于在浏览器中显示图像。ImageMap 允许在图像上定义“热点”，当单击时，要么引起一个到服务器的回发，要么导航到另一个页面。

Image 服务器控件使用户可以在 ASP.NET 网页上显示图像，并用自己的代码管理这些图像。可以在设计时或运行时以编程方式为 Image 对象指定图形文件。还可以将控件的 ImageUrl 属性绑定到一个数据源，以根据数据库信息显示图形。该控件比较特殊的一点是它不支持任何事件。

ImageMap 控件由两个元素组成。一个是图像，它可以是任何标准 Web 图形格式的图形，如.gif、.jpg 或.png 文件；另一个元素是 HotSpot(热点)控件的集合。每个热点控件都是一个类型为 CircleHotSpot、RectangleHotSpot 或 PolygonHotSpot 的不同项。对于每个热点控件，都要定义用于指定该热点的位置和大小的坐标。例如，要创建一个 CircleHotSpot 控件，则需要定义圆心的 x 和 y 坐标以及圆的半径。

每一个热点都可以是一个单独的超链接或回发事件。可以指定用户单击热点时发生的事件，可以将每个热点配置为可以转到为该热点提供的 URL 的超链接。或者，也可以将控件配置为在用户单击某个热点时执行回发。回发会引发 ImageMap 控件的 Click 事件。在事件处理程序中，可以读取分配给每个热点的唯一值。

2. AdRotator

这个控件允许在 Web 站点上显示随机广告。这些广告来自在服务器上创建的 XML 文件。每次刷新页面时都将更改显示的广告。广告可以加权以控制广告条的优先级别，这可以使某些广告的显示频率比其他广告高。也能编写在广告间循环的自定义逻辑。

该控件可显示.gif 文件或其他图形图像。当用户单击广告时，系统会将它们重定向到指定的 URL。

AdRotator 控件的所有属性都是可选的。XML 文件中可以包括下列属性。

- ImageUrl：要显示的图像的 URL。
- NavigateUrl：单击 AdRotator 控件时要跳转到的网页 URL。
- AlternateText：图像不可用时显示的文本。
- Keyword：可用于筛选特定广告的广告类别。
- Impressions：一个指示广告的可能显示频率的数值(加权数值)。在 XML 文件中，所有 Impressions 值的总和不能超过 2 048 000 000-1。
- Height：广告的高度，以像素为单位。
- Width：广告的宽度，以像素为单位。

下面的例子将使用 AdRotator 控件显示一些大学的广告信息，当用户单击广告时，系

统会导航到指定的目标 URL。

例 3-4：使用 AdRotator 控件。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框在该网站中添加一个名为 AdRotator.aspx 的 Web 窗体页。

(2) 在网站根目录下新建名为 Images 的文件夹，将准备好的广告图片复制到该文件夹中。

(3) 在“解决方案资源管理器”窗口中右击项目名称，选择“添加 ASP.NET 文件夹”| App_Data 命令，添加 App_Data 文件夹，然后右击 App_Data，从弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 3-18 所示，可以看到，这个“添加新项”对话框中可选的模板文件都是与数据文件相关的，在此选择“XML 文件”选项。

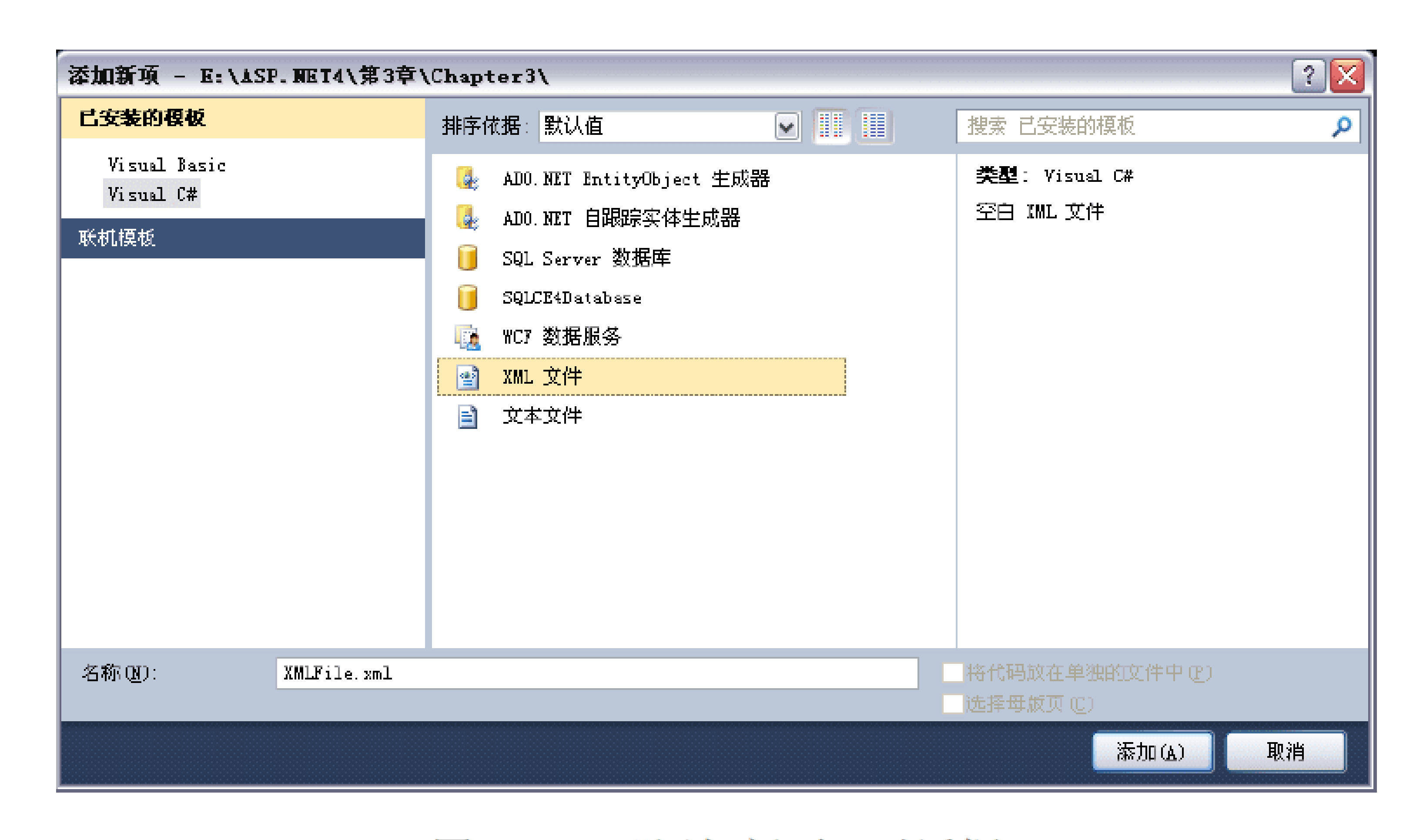


图 3-18　“添加新项”对话框

(4) 向广告信息文件中添加如下 XML 元素：

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements xmlns="http://schemas.microsoft.com/AspNet/AdRotator-Schedule-File">
 <Ad>
     <ImageUrl>~/images/tsinghua.jpg</ImageUrl>
     <NavigateUrl>http://www.tsinghua.edu.cn</NavigateUrl>
     <AlternateText>清华大学</AlternateText>
     <Impressions>100</Impressions>
 </Ad>
 <Ad>
     <ImageUrl>~/images/pku.gif</ImageUrl>
     <NavigateUrl>http://www.pku.edu.cn</NavigateUrl>
     <AlternateText>北京大学</AlternateText>
     <Impressions>50</Impressions>
 </Ad>
 <Ad>
     <ImageUrl>~/images/zju.jpg</ImageUrl>
     <NavigateUrl>http://www.zju.edu.cn</NavigateUrl>
```



```
        <AlternateText>浙江大学</AlternateText>
        <Impressions>100</Impressions>
    </Ad>
    <Ad>
        <ImageUrl>~/images/nju.jpg</ImageUrl>
        <NavigateUrl>http://www.nju.edu.cn</NavigateUrl>
        <AlternateText>南京大学</AlternateText>
        <Impressions>50</Impressions>
    </Ad>
</Advertisements>
```

(5) 打开 AdRotator.aspx 文件的“设计”视图，在网页上要显示广告的位置添加一个 AdRotator 控件，单击控件右上角的小三角，弹出“AdRotator 任务”面板，单击“选择数据源”下拉列表，选择“<新建数据源>”选项，如图 3-19 所示。

(6) 在打开的“数据源配置向导”对话框中选择“XML 文件”数据源，如图 3-20 所示。



图 3-19　“AdRotator 任务”面板

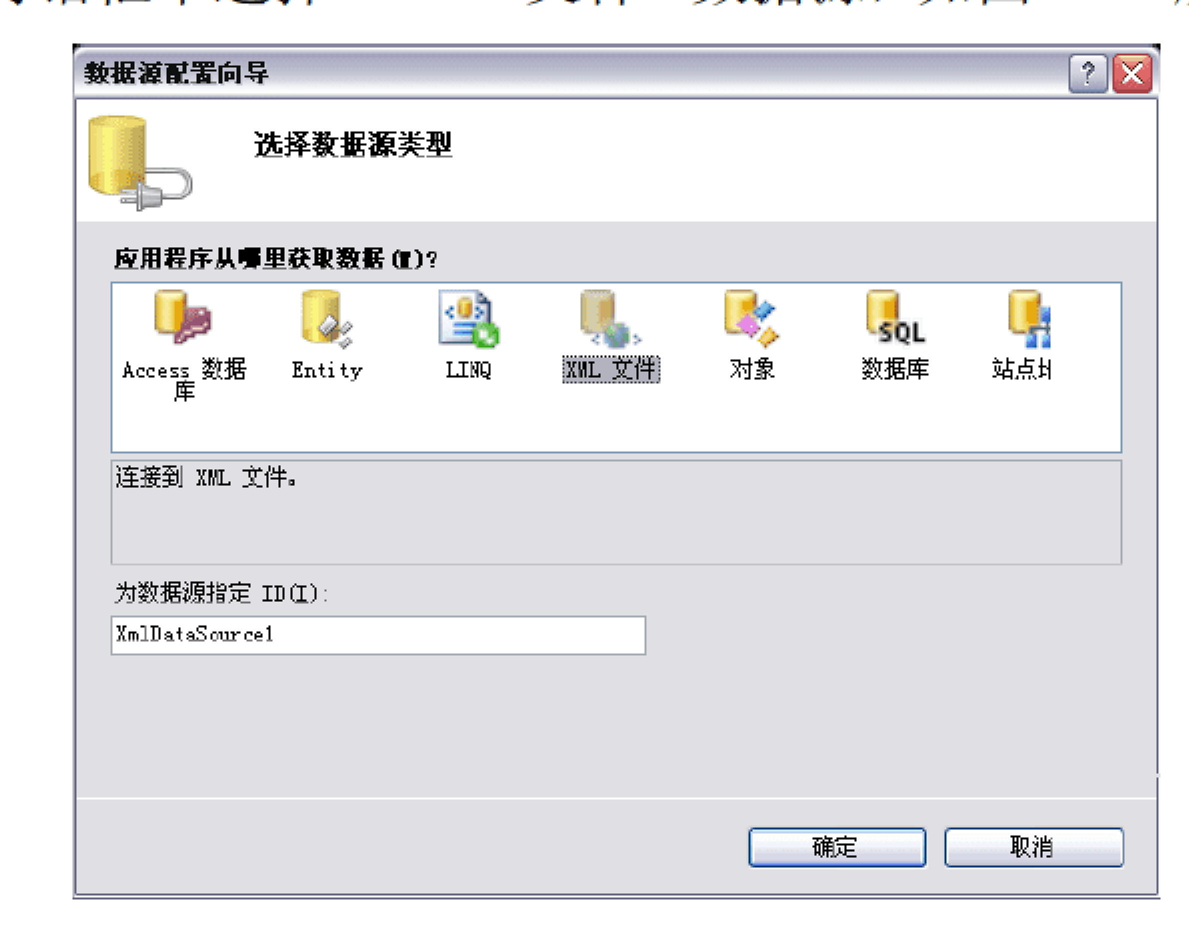


图 3-20　“数据源配置向导”对话框

(7) 单击“确定”按钮，打开“配置数据源”对话框，单击“数据文件”后面的“浏览”按钮，选择刚才创建的 XML 文件，如图 3-21 所示。



图 3-21　“配置数据源”对话框

(8) 编译并运行程序，在浏览器中打开 AdRotator.aspx 页面，按 F5 键或者单击工具栏的“刷新”按钮，刷新页面，可以看到页面中将显示不同的图片。

3. Calendar

Calendar 控件提供了一个功能丰富的接口，允许用户选择日期。

4. FileUpload

FileUpload 控件允许用户上传可以存储在服务器上的文件。

5. HiddenField

HiddenField 控件可用来将数据存储在各个请求提交的页面中。如果希望页面记住特定数据，而用户在页面中又不会看到，那么该控件就很有用。由于这个字段会显示在页面的 HTML 源代码中，因此终端用户可以访问，所以不要在其中存储任何敏感数据。

6. Literal、Localize 和 Substitute

这 3 个控件看起来有些像 Label 控件，因为它们都可以显示静态文本或 HTML。Literal 最大的优点是它本身不呈现额外的标记；它仅显示赋予 Text 属性的信息，因此对于显示 HTML 或者显示在 Code Behind 中构建的或从数据库检索的 JavaScript 非常有用。

Literal 控件常用的属性是 Mode 属性，该属性用于指定控件对用户所添加的标记的处理方式。可以将 Mode 属性设置为：Transform(将对添加到控件中的任何标记进行转换，以适应请求浏览器的协议)、PassThrough(添加到控件中的任何标记都将按原样呈现在浏览器中)和 Encode(使用 HtmlEncode 方法对添加到控件中的任何标记进行编码，这会将 HTML 编码转换为其文本表示形式)。

Localize 控件用在使用多种语言的 Web 站点中，并且能够从翻译后的资源文件中检索其内容。Substitute 控件用在高级缓存场景中，并且允许仅更新部分没有完全缓存的页面。

7. Table

<asp:Table>控件在很多方面等同于 HTML<table>元素。然而，由于该控件位于服务器上，因此可以对它进行编程，动态地创建新的列和行，以及向其中添加动态数据。

8. XML

XML 控件允许将数据从 XML 格式转换为另一种格式(如 XHTML)，以便显示在页面上。

3.3　HTML 控件

工具箱的 HTML 类别中包含许多 HTML 控件，它们看起来与标准类别中的控件很相似。例如，Input (Button)控件看起来就很像<asp:Button>。类似地，Select 控件有<asp:DropDownList>和<asp:ListBox>作为它的对应控件。

在 VWD 2010 中，从工具箱添加到页面上的 HTML 控件只是已设置了某些属性的 HTML 元素，当然也可通过输入 HTML 标记在“源”视图中创建 HTML 元素。

3.3.1　HTML 元素

默认情况下，ASP.NET 文件中的 HTML 元素将被作为文本进行处理，并且不能在服务器端代码中引用这些元素，只能在客户端通过 javascript 和 vbscript 等脚本语言来控制。

工具箱的 HTML 选项卡上提供了一些基于 HTML INPUT 元素的控件。常用的 HTML INPUT 元素是 Input (Button)控件和 Input (Text)控件。

Input(Button)元素的主要功能是创建一个用来触发事件处理程序的按钮，通过使用 onclick 属性来表明单击按钮可以触发的处理方法。其主要属性如下。

- ID：此控件的编程名称。
- value：设置按钮中显示的文字。
- 按钮控件：默认情况下是“INPUT type="button"元素”。

文本框控件：默认情况下是“INPUT type="text"元素”。

Input(Text)元素创建允许用户在其中输入文本或密码的单行文本框，其主要属性如下。

- Type=text/password：文本框的类型。
- MaxLength：文本框中最大的输入字节。
- Size：设定文本框的宽度。
- Value：设定文本框的值。

3.3.2　HTML 服务器控件

默认情况下，ASP.NET 文件中的 HTML 元素被视为传递给浏览器的标记，作为文本进行处理，不能在服务器端的代码中引用这些元素。若要使这些元素能以编程方式进行访问，可以通过添加 runat="server"属性表明应将 HTML 元素作为服务器控件进行处理，这样就可使用基于服务器的代码对其进行编程引用了。

添加了属性 runat="server"的 HTML 元素就转换为 HTML 服务器控件了。移除控件标记中的 runat="server"属性，HTML 服务器控件就转换为 HTML 元素。

工具箱的 HTML 类别中就是 HTML 服务器控件，标准控件和 HTML 控件之间似乎有一些重叠，但是 HTML 控件的功能比标准类别中的控件的功能少得多。一般来说，标准类别中的真正服务器控件提供了更多的功能，无论是在 VWD 中的设计时支持方面还是在运行时能做的事情方面都是如此。不过这种功能是有代价的。因为它们增加了复杂度，所以处理服务器控件会多花一点时间。然而，在大多数 Web 站点上，可能不会注意到这一差别。只有在高通信量的 Web 站点，且在页面上有很多控件时，使用 HTML 控件才会提供稍好一些的性能。

说明：

在大多数情况下，人们更愿意使用服务器控件而不是与它们对应的 HTML 控件。因为服务器控件提供了更多的功能，在页面中更灵活，可以给用户带来更丰富的体验。而且有比较好的设计时支持，因此值得选择。如果十分确信不需要服务器控件提供的这些功能，则可以选择 HTML 控件。

3.4　验证控件

通常，Internet 上的多数 Web 站点都要处理来自用户的输入。为了防止系统接收无效数据，在允许系统使用之前要先验证数据的有效性。另一方面，一旦站点发布到 Internet 上，就可能会成为恶意用户和黑客的入侵目标。因此，在将数据用于应用程序之前验证数据的有效性格外重要。

ASP.NET4 为开发人员提供了一套完整的服务器控件来验证用户输入的信息是否有效，这些控件可与 ASP.NET 网页上的任何控件(包括 HTML 和服务器控件)一起使用。

3.4.1　验证控件简介

ASP.NET 提供的 6 个有效性验证控件中，5 个控件用来执行实际的有效性验证，而最后一个控件 ValidationSummary 用来向用户提供页面中出现的错误的反馈信息。

有效性验证控件最出色的是它们能在客户端和服务器上检查输入。当向 Web 页面中添加一个有效性验证控件时，控件就会呈现在客户端验证关联控件有效性的 JavaScript。大多数启用了 JavaScript 的现代 Web 浏览器(包括 IE、Firefox、Chrome、Opera 和 Safari)都能进行这种客户端有效性验证。同时，有效性验证也可以在服务器上自动进行。这样就容易向用户提供关于使用客户端脚本的数据的即时反馈，从而使 Web 页面在服务器上免受伪数据的侵扰。

如表 3-4 所示列出了 ASP.NET 提供的验证控件及其功能说明。

表 3-4　ASP.NET 验证控件

验 证 类 型	使用的控件	说　　明
必选项	RequiredFieldValidator	验证一个必填字段，如果该字段没填，那么将不能提交信息
与某值的比较	CompareValidator	将用户输入与一个常数值或者另一个控件或特定数据类型的值进行比较(使用小于、等于或大于等比较运算符)，同时也可以用来校验控件中内容的数据类型，如整形、字符串型等。如密码和确认密码两个字段是否相等
范围检查	RangeValidator	RangeValidator 控件可以用来判断用户输入的值是否在某一特定范围内。可以检查数字对、字母对和日期对限定的范围。属性 MaximumValue 和 MinimumValue 用来设定范围的最大和最小值
模式匹配	RegularExpressionValidator	它根据正则表达式来验证用户输入字段的格式是否合法，如电子邮件、身份证和电话号码等。ControlTo-Validate 属性选择需要验证的控件，ValidationExpression 属性则编写需要验证的表达式的样式

(续表)

验 证 类 型	使用的控件	说　　明
用户定义	CustomValidator	使用自己编写的验证逻辑检查用户输入。此类验证能够检查在运行时派生的值。在运行定制的客户端 JavaScript 或 VBScript 函数时，可以使用这个控件
验证汇总	ValidationSummary	该控件不执行验证，但该控件会将本页所有验证控件的验证错误信息汇总为一个列表并集中显示，列表的显示方式由 DisplayMode 属性设置

1. 验证控件的共有属性

前 5 个验证控件基本上都继承自同一个基类，因此它们有一些共同的行为，5 个有效性验证控件中的 4 个以相同的方式操作，并包含允许验证关联控件的内置行为，CustomValidator 控件则允许用户编写非内置的自定义功能。如表 3-5 所示为有效性验证控件共有的常用属性。

表 3-5　有效性验证控件的共有属性

属　　性	说　　明
Display	该属性确定隐藏的错误消息是否占用空间。如果将 Display 设置为 Static，错误消息就会占用屏幕空间，即使在隐藏时也是如此。如果设置为 None，则看不到错误消息
CssClass	这个属性允许设置应用到错误消息文本的 CssClass 特性
ErrorMessage	这个属性保存用在 ValidationSummary 控件中的错误消息。当 Text 属性为空时，也用 ErrorMessage 值作为出现在页面上的文本
Text	Text 属性用作有效性验证控件显示在页面上的文本。它可以是一个星号(*)以表示出现一个错误，也可以是具体的文本信息
ControlToValidate	这个属性包含需要验证有效性的控件的 ID
EnableClientScript	这个属性用于确定控件是否提供客户端的有效性验证，默认为 True
SetFocusOnError	这个属性确定客户端脚本是否将焦点放在产生错误的第一个控件上，默认值为 False
ValidationGroup	有效性验证控件可以组合在一起，允许对选中的控件进行有效性验证。同一个 ValidationGroup 中的所有控件都会被同时检查，这意味着如果控件不是这个控件组的一部分，就不对它进行有效性验证
IsValid	通常在设计时不会设置这个属性，不过在运行时它提供关于是否通过了有效性验证测试的信息

乍一看，Text 和 ErrorMessage 属性的作用似乎是一样的。它们都可以用来以错误消息的形式向用户提供反馈。但是，当与 ValidationSummary 控件结合起来使用时，两者之间

就有了细微的区别。当同时设置这两个属性时，Validation 控件显示 Text 属性，而 ValidationSummary 控件则显示 ErrorMesage 属性。

2. RangeValidator 控件

RangeValidator 控件允许检查一个值是否落在特定的范围内。这个控件能检查字符串、数字、日期和货币等数据类型。除了上述共有属性之外，RangeValidator 控件还有其他几个重要属性，如表 3-6 所示。

表 3-6　RangeValidator 控件的重要属性

属　性	说　　明
MinimumValue	该属性确定可接受的最小值。例如，当检查 1 和 10 之间的整数时，将该属性设置为 1
MaximumValue	该属性确定可接受的最大值。例如，当检查 1 和 10 之间的整数时，将该属性设置为 10
Type	该属性确定有效性验证控件检查的数据类型。可以设置为 String、Integer、Double、Date 或 Currency 来检查各自的数据类型

3. CompareValidator 控件

CompareValidator 控件能用来比较一个控件的值与另一个控件的值。它通常用在注册表单中，用户必须输入密码两次，以确保两次输入的密码相同。也可以不与另一个控件作比较，而是与一个常量值比较。CompareValidator 控件的其他属性如表 3-7 所示。

表 3-7　CompareValidator 控件的其他属性

属　性	说　　明
ControlToCompare	该属性包含验证器要与之比较的控件 ID。设置了该属性，ValueToCompare 就无效了
Operator	该属性确定比较操作的类型。如，当 Operator 设置为 Equal 时，两个控件都必须包含验证器认为有效的同一个值。类似地，还有一些选项，如 NotEqual、GreaterThan 和 GreaterThanEqual，用来执行不同的有效性验证操作
Type	该属性确定有效性验证控件检查的数据类型。可以设置为 String、Integer、Double、Date 或 Currency 来检查各自的数据类型
ValueToCompare	该属性允许定义一个要比较的常量值。它通常用在必须输入 Yes 这样的单词的协议中，表示同意某些条件。只要将 ValueToCompare 设置为单词 Yes，并将 ControlToValidate 设置为要验证有效性的控件，就可以了

4. CustomValidator 和 ValidationSummary 控件

CustomValidator 控件允许为客户端(用 JavaScript)和服务器(用 VB.NET 或 C#)编写自定

义的有效性验证函数。这样在要验证有效性的数据和要应用的规则方面就有了相当大的灵活性。

ValidationSummary 控件向用户提供了它从单个有效性验证控件的 ErrorMessage 属性中检索到的一个错误列表。它能以 3 种不同的方式显示这些错误，使用一个嵌在页面中的列表、使用 JavaScript 警报框或者同时使用这两种方式。可以通过 ShowMessageBox 和 ShowSummary 属性控制这个设置。此外，DisplayMode 属性可以修改表现错误列表的方式。默认设置为 BulletList，其中每个错误都是项目列表中的一个项。

3.4.2　使用验证控件

验证控件的使用非常简单，只需将他们添加到页面，设置一些属性即可。

例 3-5：使用验证控件。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框在该网站中添加一个名为 Validator.aspx 的 Web 窗体页。

(2) 打开 Validator.aspx 的“设计”视图，选择“表”|“插入表”命令打开“插入表格”对话框，插入一个 9 行 3 列的表格，如图 3-22 所示。

(3) 合并第一行的 3 个单元格。首先，选中这 3 个单元格，然后选择“表”|“修改”|“合并单元格”命令即可。

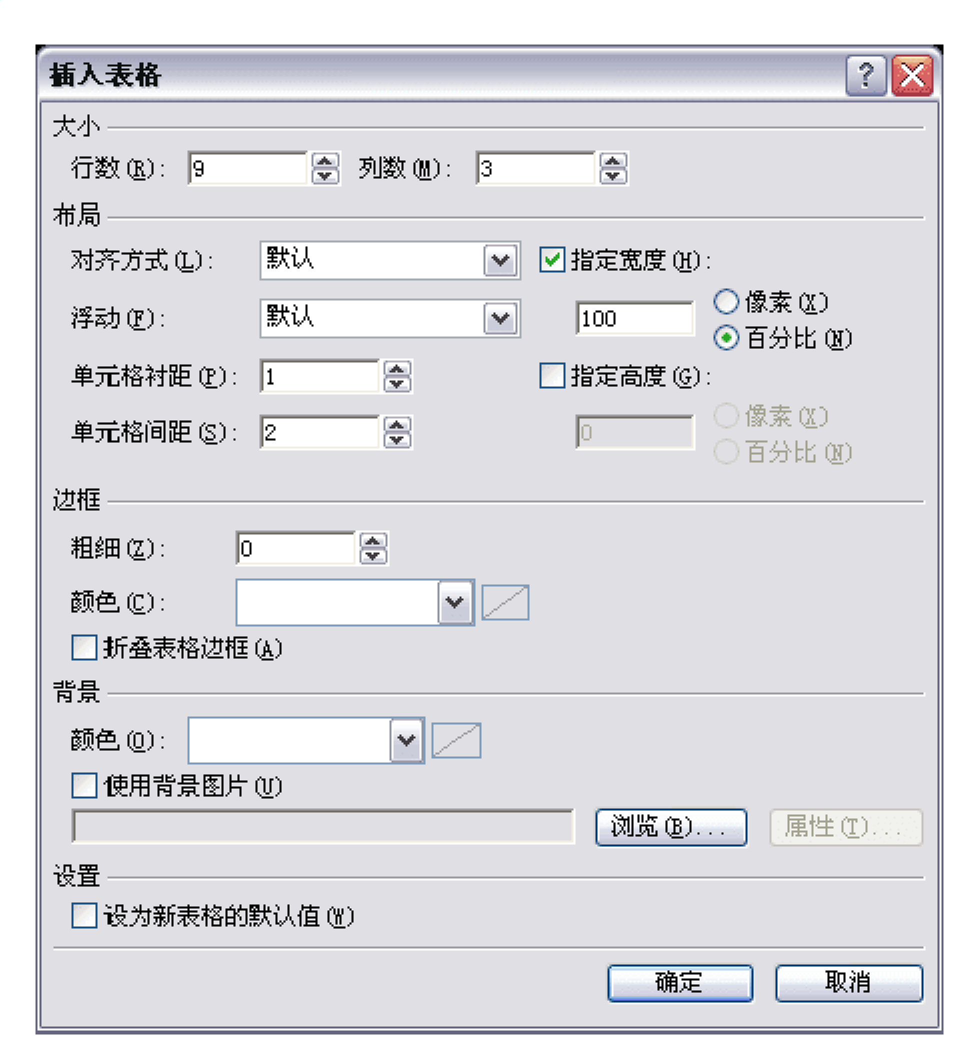


图 3-22　“插入表格”对话框

(4) 在第一行的单元格中输入文本“注册表单”。在第 2~8 行的单元格中，第 1 列单元格中输入文本信息，在第 2 列单元格中添加用于输入信息的 TextBox 控件，第 3 列单元格中添加相应的验证控件。在最后一行中，第 1 列添加一个 Button 控件，用于提交表单数据，第 2 列添加一个 Label 控件，用于显示提示信息，第 3 列添加一个 ValidationSummary 控件显示验证错误信息。如图 3-23 所示。

(5) TextBox 控件从上至下，ID 依次为 TextBox1~TextBox7。设置 TextBox4 和 TextBox5 的 TextMode 属性为 Password。

(6) 同时选中 5 个 RequireFieldValidator 控件，设置其 Text 属性为“*”，并且分别设置它们的 ErrorMessage 属性为“姓名不能为空”、“年龄不能为空”、“Email 不能为空”、“密码不能为空”和“确认密码不能为空”。设置每个 RequireFieldValidator 控件的 ControlToValidator 属性为其所在行的 TextBox 控件的 ID。

(7) 设置 RangeValidation 控件的 ControlToValidator 属性为 TextBox2，Minimum Value 属性为 1，Maximum 属性为 100，Type 属性为 Integer，ErrorMessage 属性为“年龄是 1~100 之间的整数”。

(8) 设置 RegularExpressionValidation 控件的 ControlToValidator 属性为 TextBox3，ErrorMessage 属性为“Email 格式错误”；然后单击 ValidationExpression 属性右边的浏览按钮，在弹出的“正则表达式编辑器”对话框中选择“Internet 电子邮件地址”选项，如图 3-24 所示。

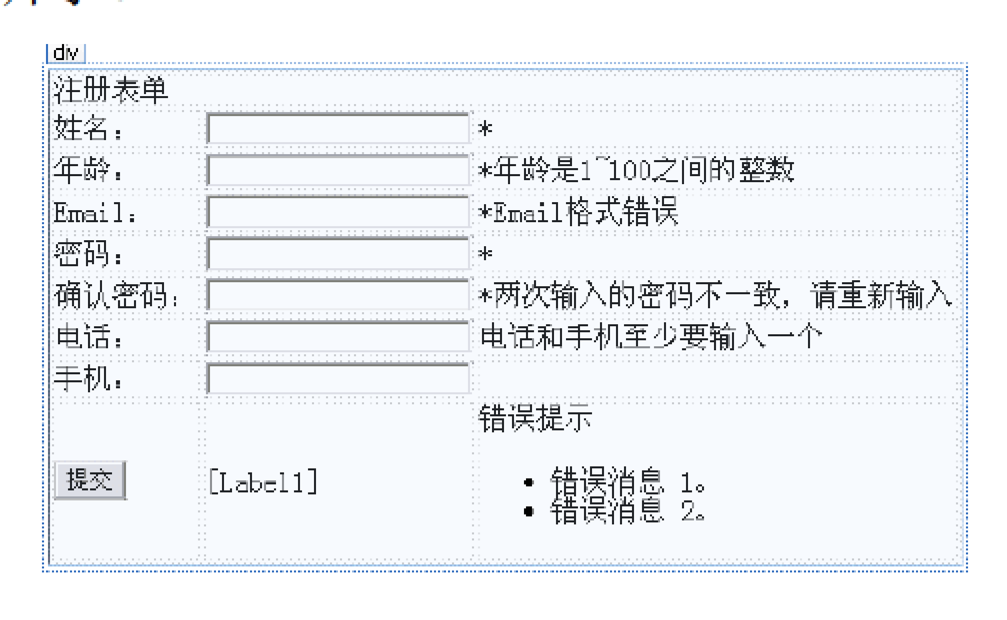


图 3-23　在表格中添加文本信息和控件



图 3-24　“正则表达式编辑器”对话框

(9) 设置 CompareValidation 控件的 ControlToValidator 属性为 TextBox5，ControlToCompare 属性为 TextBox4，ErrorMessage 属性为“两次输入的密码不一致，请重新输入”。

(10) “电话”行对应的验证控件是 CustomValidator 控件，设置该控件的 Display 属性为 Dynamic，ErrorMessage 属性为“电话和手机至少要输入一个”，然后为该控件添加 ServerValidate 事件处理程序。代码如下：

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (!string.IsNullOrEmpty(TextBox6.Text) ||!string.IsNullOrEmpty(TextBox7.Text))
    {
        args.IsValid = true;
    }
    else
    {
        args.IsValid = false;
    }
}
```

(11) 设置 CustomValidator 控件的 ClientValidationFunction 属性为 ValidatePhone，切换到页面的“源”视图，在<body>之前添加如下 JavaScript 代码：

```
<script type="text/javascript">
    function ValidatePhone(source, args) {
        var telephone = document.getElementById('<%= TextBox6.ClientID %>');
        var mobile = document.getElementById('<%= TextBox7.ClientID %>');
        if (telephone.value != '' || mobile.value != '') {
            args.IsValid = true;
        }
        else {
            args.IsValid = false;
        }
    }
</script>
```

上述 JavaScript 函数 ValidatePhoneNumbers 是客户端验证函数，确保在将页面提交回服务器之前至少输入了一个电话号码。

(12) 设置按钮控件的 Text 属性为“提交”，Label 控件的 Text 属性为空。为按钮控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "所有输入信息验证通过！";
}
```

(13) 编译并运行程序，在浏览器中加载页面 Validator.aspx，如果没有输入任何信息，或者输入信息不合法，将给出错误提示，如图 3-25 所示。

(14) 回到 VWD 中，将 ValidationSummary 控件的 ShowMessageBox 设置为 True，ShowSummary 设置为 False。再次编译并运行程序，在浏览器中打开页面，注意此时得到的不是含有错误的内联列表，而是一个警告对话框，如图 3-26 所示。

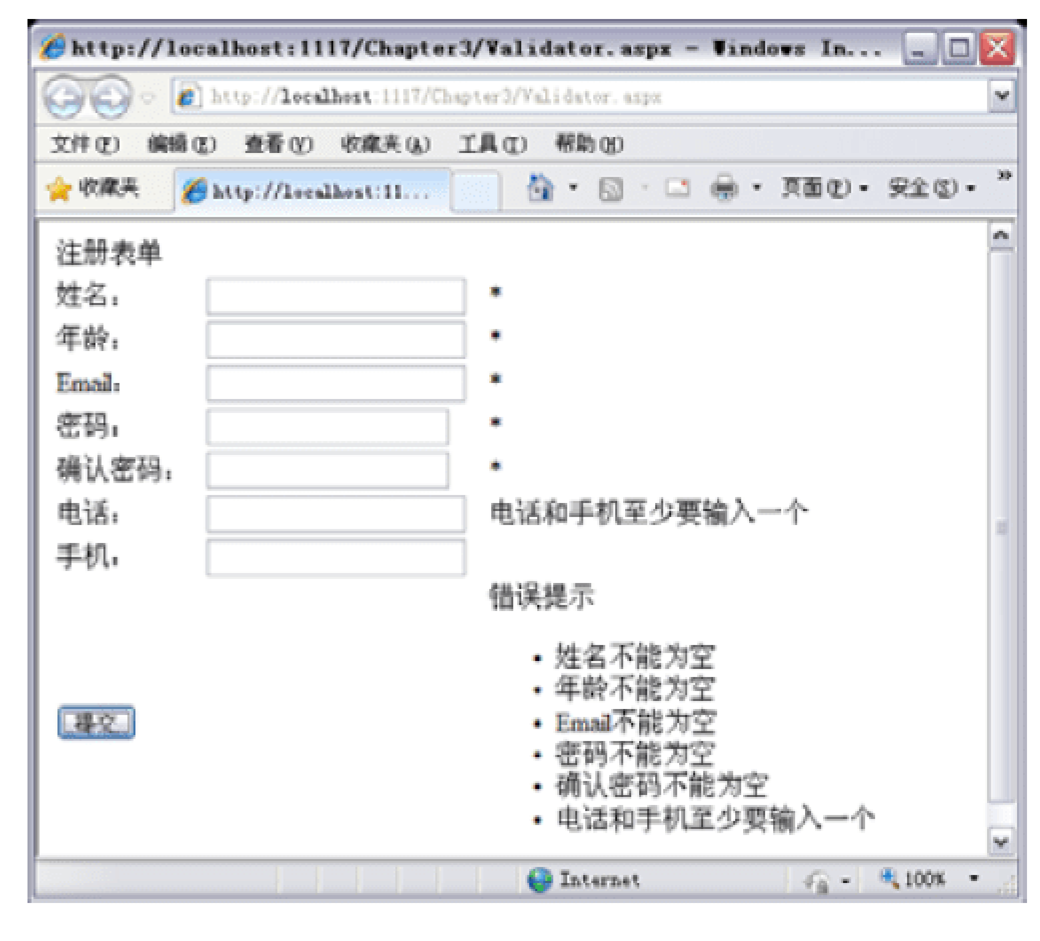


图 3-25　页面验证效果

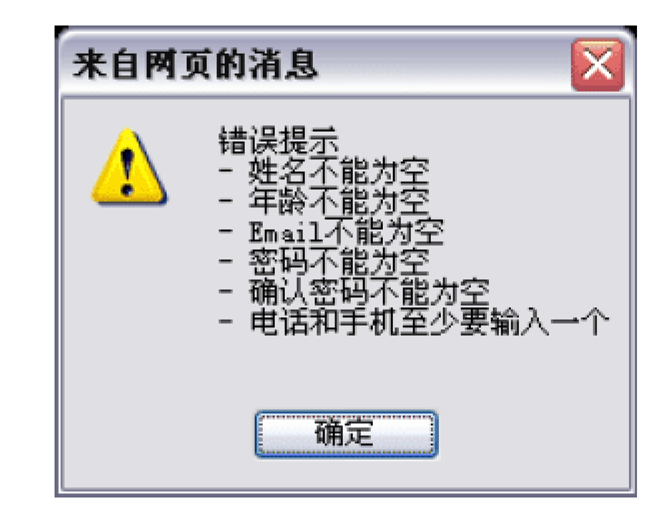


图 3-26　以对话框形式显示错误

(15) 如果信息输入全部正确，则全部验证通过，如图 3-27 所示。

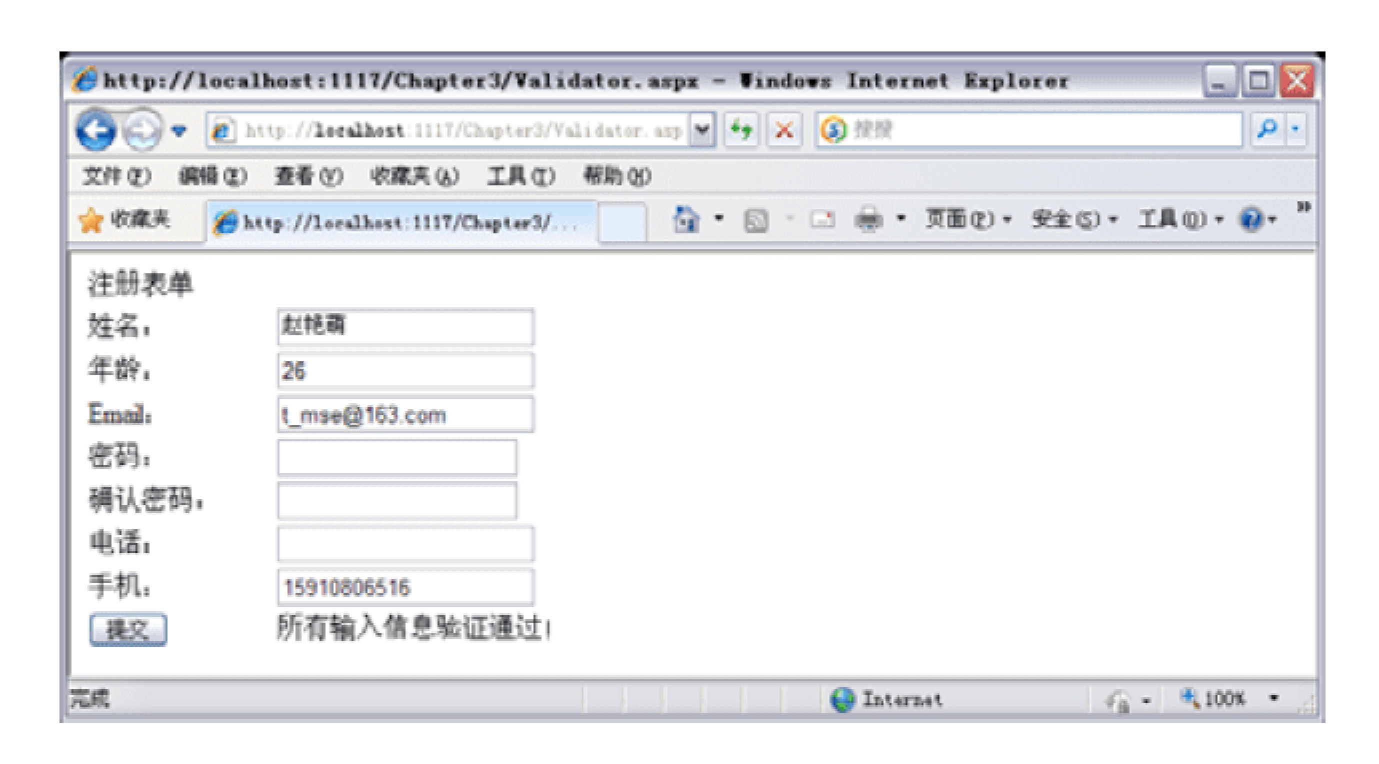


图 3-27　全部验证通过

3.5　导航控件

当站点包含的页面比较多时，有一个稳固而清晰的导航结构就很重要，这样才能让用户顺畅地浏览站点。使用良好的导航系统，项目中所有没有连接的 Web 页面就会形成一个完整而连贯的 Web 站点。

ASP.NET 4 提供了 3 个有用的导航控件：SiteMapPath、TreeView 和 Menu。

- SiteMapPath：这个 Web 控件提供一个面包条(breadcrumb)，它是一行文本，显示用户当前在网站结构中的位置。例如，在网上书店中，用户浏览到《Visual C++》时，面包条可能类似于“主页->计算机->编程类->Visual C++”，其中每部分(如主页，计算机等)都显示为返回到前一部分的链接。面包条能够让用户快速地查看当前在网站中的位置，并沿逻辑层次结构向上导航。
- Menu：这个 Web 控件提供网站结构的层次视图。对于学校的网站，顶层菜单将包含主类别(如学校介绍、机构设置和新闻等)，每个菜单项又可以包含各自的子菜单，显示各自的子类别。
- TreeView：树视图提供了与菜单相同的数据，惟一的区别是显示数据的方式。树视图显示为可展开或可折叠的树，而菜单(Menu)是由菜单项和子菜单组成。

一般情况下，开发人员利用站点地图和 SiteMapPath 控件实现自动导航，利用 Menu 控件或者 TreeView 控件实现自定义导航。

3.5.1　创建站点地图

为了更容易地使 Menu、TreeView 或 SiteMapPath 显示站点中的相关页面，ASP.NET 使用一个基于 XML 的文件来描述 Web 站点的逻辑结构。默认情况下，这个文件名为 Web.sitemap。站点中的导航控件会用这个文件以有组织的方式表现相关的链接。只要将一个导航控件与这个 Web.sitemap 文件挂钩，就能创建复杂的用户界面元素，如折叠菜单或树型视图等。

注意：

默认情况下，应将站点地图文件命名为 Web.sitemap。这样控件就可以自动找到正确的文件。对于更高级的情况，可以有多个不同名称的站点地图文件，且在向系统提供这些附加文件的 web.config 中有一个配置设置。在大多数情况下，有一个站点地图文件就足够了。

站点地图文件的框架结构如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode url="~/" title="首页" description="Go to the homepage">
    <siteMapNode url="~/Info.aspx" title="概述" description="Reviews published on this site" />
    <siteMapNode url="~/About.aspx" title="关于"   description="About this site" />
  </siteMapNode>
</siteMap>
```

这个站点地图文件的根节点是 siteMap，其下使用 siteMapNode 节点建立层次结构，每个 siteMapNode 可以有多个子节点(但是，在 siteMap 元素下只能有一个 siteMapNode)，子节点仍然用<siteMapNode>定义。可以用来创建一个既有广度又有深度的站点结构。在本例中，只有一个名为“首页”的根节点，它包含两个子元素：“概述”和“关于”。本例中的 siteMapNode 元素有 3 个特性集：url、title 和 description。

- url 特性应指向 Web 站点中的有效页面。可以用~语法来引用基于应用程序根文件夹的 URL。此 URL 在网站地图中必须唯一。
- title 特性用于显示页面的名称。在使用 Menu、TreeView 和 SiteMapPath 控件时将看到关于它的更多信息。
- description 特性用作导航元素的工具提示，可有可无。

技巧：

虽然 ASP.NET 运行库不允许多次指定同一个 URL，但是可以通过添加一个查询字符串使 URL 唯一来绕过这一问题。例如，~/Login.aspx 和~/Login.aspx?name=gemm 会被看作两个不同的页面。

VWD 没有自动基于当前站点的结构创建站点地图文件的方式。要创建一个有用的 Web.sitemap 文件，需要向站点中添加一个文件，然后手动向它添加必需的 siteMapNode 元素。

例 3-6：创建站点地图文件 Web.sitemap。

(1) 启动 VWD 2010，打开网站 Chapter3，打开“添加新项”对话框，选择“站点地图”选项，添加默认名称为 Web.sitemap 的站点地图文件。

(2) 新添加的 Web.sitemap 文件中会出现一个包含两个子节点的根元素，修改 Web.sitemap 文件使它包含下列代码：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
```

```
<siteMapNode url="~/Default.aspx" title="首页"  description="首页">
    <siteMapNode url="~/Hall.aspx" title="大厅"  description="广播大厅" >
    </siteMapNode>
    <siteMapNode url="~/GuangChang.aspx" title="微薄广场"  description="微薄广场" >
        <siteMapNode url="~/YuLe.aspx" title="娱乐空间"  description="娱乐空间" />
        <siteMapNode url="~/XinLing.aspx" title="心灵天地"  description="心灵天地" />
    </siteMapNode>
</siteMapNode>
</siteMap>
```

(3) 保存文件，完成站点地图的创建。

Web.sitemap 文件本身用处并不大。需要向站点中添加导航控件来使用站点地图。下面将介绍如何使用导航控件来实现网站导航。

3.5.2 使用 SiteMapPath 控件

定义好站点地图之后，就可以使用 SiteMapPath 控件显示导航路径了，也就是显示当前页面在网站中的位置。只需将该控件拖放到站点地图中包含的.aspx 页面上，它就会自动实现导航，不需要开发者编写任何代码。

为了能够使用 Web.sitemap 文件，ASP.NET 使用了 SiteMapDataSource 控件，这是一个数据控件。当使用 SiteMapPath 控件显示"痕迹导航"时，ASP.NET 会自动找到 Web.sitemap 文件。使用另外两个导航控件时，则需要显式地指定一个 SiteMapDataSource 作为 Web.sitemap 文件的中间层。

添加站点地图到应用程序中时，需要将站点地图放在 Web 应用程序的根目录下，并保持其文件为 Web.sitemap。如果将该文件放在另一个文件夹中或选择不同的文件名，SiteMapPath 导航控件将不能找到站点地图，就不能知道网站的结构，因为默认情况下 SiteMapPath 导航控件在根目录下寻找名为 Web.sitemap 的文件。

SiteMapPath 控件显示了当前在站点结构中的位置。它将自身表现为一系列链接，常称之为痕迹导航(breadcrumb)。它是一个非常简单但功能强大的控件，有 50 多个公有属性，可以通过"属性"面板来设置这些属性。

像大多数 Web 控件一样，SiteMapPath 控件也有很多可用于定制其外观的属性。如表 3-8 所示为 SiteMapPath 控件的常用属性。

表 3-8　SiteMapPath 控件的常用属性

属　　性	说　　明
CurrentNodeStyle	定义当前节点的样式，包括字体、颜色和样式等
NodeStyle	定义导航路径上所有节点的样式
ParentLevelsDisplayed	指定在导航路径上显示的相对于当前节点的父节点层数。默认值为-1，表示父级别数没有限制

(续表)

属　　性	说　　明
PathDirection	指定导航路径上各节点的显示顺序。默认值为 RootToCurrent，即按从左到右的顺序显示从根节点到当前节点的路径。另一选项为 CurrentToRoot，即按相反的顺序显示导航路径
PathSeparator	指定导航路径中节点之间分隔符。默认值为>，也可自定义为其他符号
PathSeparatorStyle	定义分隔符的样式
RenderCurrentNodeAsLink	是否将导航路径上当前页名称显示为超链接。默认值为 false
RootNodeStyle	定义根节点的样式
ShowToolTips	当鼠标悬停于导航路径的某个节点时，是否显示相应的工具提示信息。默认值为 true，即当鼠标悬停于某节点上时，显示该节点在站点地图中定义的 Description 属性值

例 3-7：使用 SiteMapPath 控件和例 3-6 中的站点地图文件实现自动导航。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框，分别添加名为 Hall.aspx、GuangChang.aspx、YuLe.aspx 和 XinLing.aspx 的网页。

(2) 在每个页面中都添加一个 SiteMapPath 控件，在“设计”视图中可以看到该页面的导航路径，如图 3-28 所示为 XinLing.aspx 的“设计”视图效果。可见，利用站点地图和 SiteMapPath 控件实现自动导航非常方便。

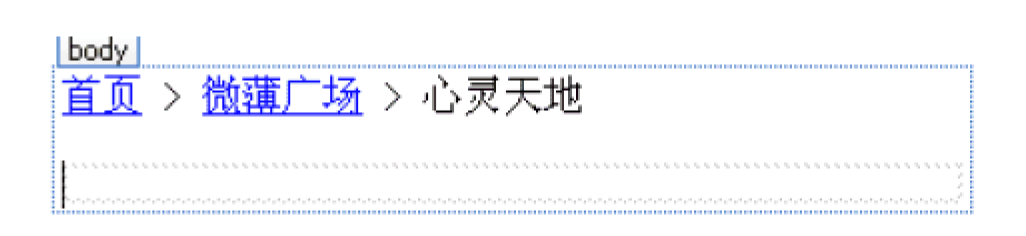


图 3-28　添加 SiteMapPath 控件后的效果

(3) 本例无须编写任何代码，编译并运行程序，在浏览器中打开不同的页面，用户体验导航的便捷。

如果要在客户端查看源文件，可以看到，SiteMapPath 呈现为一系列包含一个链接或纯文本的<span>元素。

3.5.3　使用 Menu 控件

Menu 控件主要用于创建一个菜单，让用户快速选择不同页面，从而完成导航功能。该控件可以包含一个主菜单和多个子菜单。菜单有静态和动态两种显示模式。静态显示模式是指定义的菜单始终完全显示，动态显示模式是指需要用户将鼠标停留在菜单项上时才显示子菜单。

Menu 控件的常用属性如表 3-9 所示。

表 3-9　Menu 控件的常用属性

属　性	说　明
DynamicEnableDefaultPopOutImage StaticEnableDefaultPopOutImage	是否在菜单各项之间显示分隔图像。默认值为 true
DynamicPopOutImageUrl StaticPopOutImageUrl	设置菜单中自定义分隔图像的 URL
DynamicBottomSeparatorImage Url StaticBottomSeparatorImageUrl	指定在菜单项下方显示图像的 URL。默认值为空字符串("")，即菜单项下方不显示任何图像
DynamicTopSeparatorImageUrl StaticTopSeparatorImageUrl	指定在菜单项上方显示图像的 URL。默认值为空字符串("")，即菜单项上方不显示任何图像
DynamicHorizontalOffset StaticHorizontalOffset	指定菜单相对于其父菜单的水平距离，单位是像素，默认值为 0。该属性值可正可负，为负值时，各菜单之间的距离会缩小
DynamicVerticalOffset StaticVerticalOffset	指定菜单相对于其父菜单项的垂直距离
MaximumDynamicDisplayLevels	设置动态菜单的最大层数。默认值为 3
Orientation	设置菜单的展开方向。有 Horizontal 和 Vertical 两个选项，默认值为 Vertical，即垂直方向
Items	获取包含 Menu 控件中的所有菜单项，返回 MenuItemCollection 对象
DataSourceID	SiteMapDataSource 控件的 ID，为 Menu 控件提供数据创建动态菜单
RenderingMode	ASP.NET 4.0 中新增的属性。这个属性用于确定控件是使用表和内联样式，还是使用无序列表和 CSS 样式来显示自身
IncludeStyleBlock	ASP.NET 4.0 中新增的属性。这个属性使得开发人员可以完全控制控件的样式

Menu 控件的用法非常灵活，设计者可以利用它定义各种菜单样式，实现类似于 Windows 窗口菜单的功能。

例 3-8：使用 Menu 控件在网页中添加一个菜单，实现自定义导航功能。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框，分别添加名为 Menu.aspx、One.aspx、One.aspx、Two1.aspx、Two2.aspx、Three1.aspx 和 Three2.aspx 的网页。

(2) 打开 Menu.aspx 文件的“设计”视图，添加一个 Menu 控件。

(3) 将 Menu 控件的 Orientation 属性设置为 Horizontal，以便使其横向排列。

(4) 单击 Menu 控件右上方的小三角，打开“Menu 任务”面板，选择“编辑菜单项”命令，如图 3-29 所示。之后将打开“菜单项编辑器”对话框，如图 3-30 所示。在此可输入各级菜单项。



图 3-29　“Menu 任务”面板

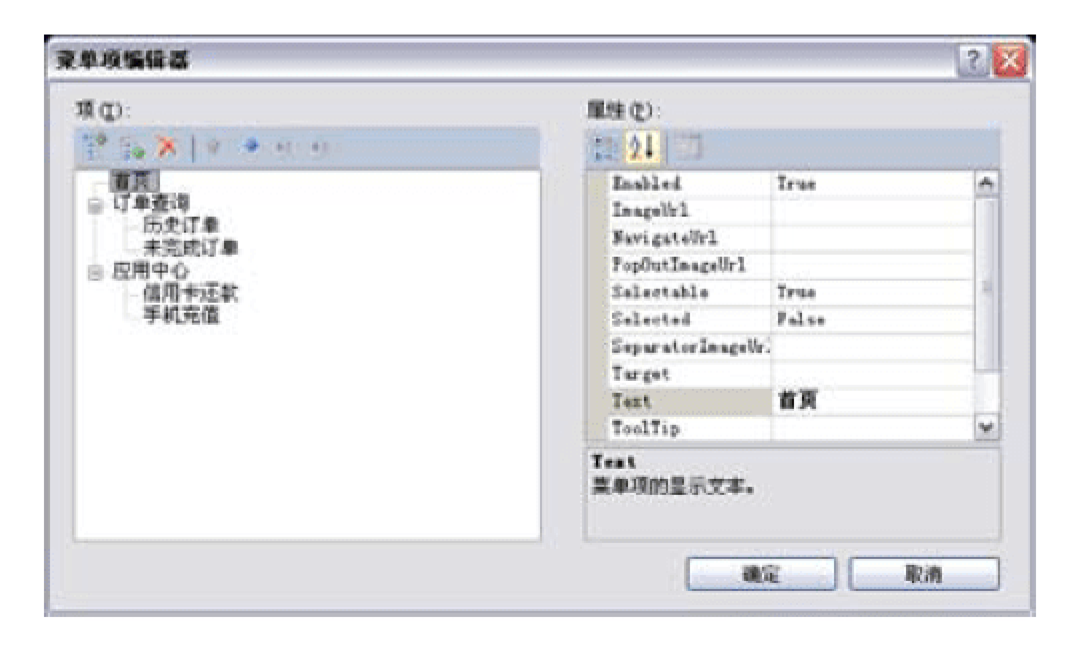


图 3-30　“菜单项编辑器”对话框

(5) “菜单项编辑器”对话框中的工具栏按钮的功能介绍如图 3-31 所示。

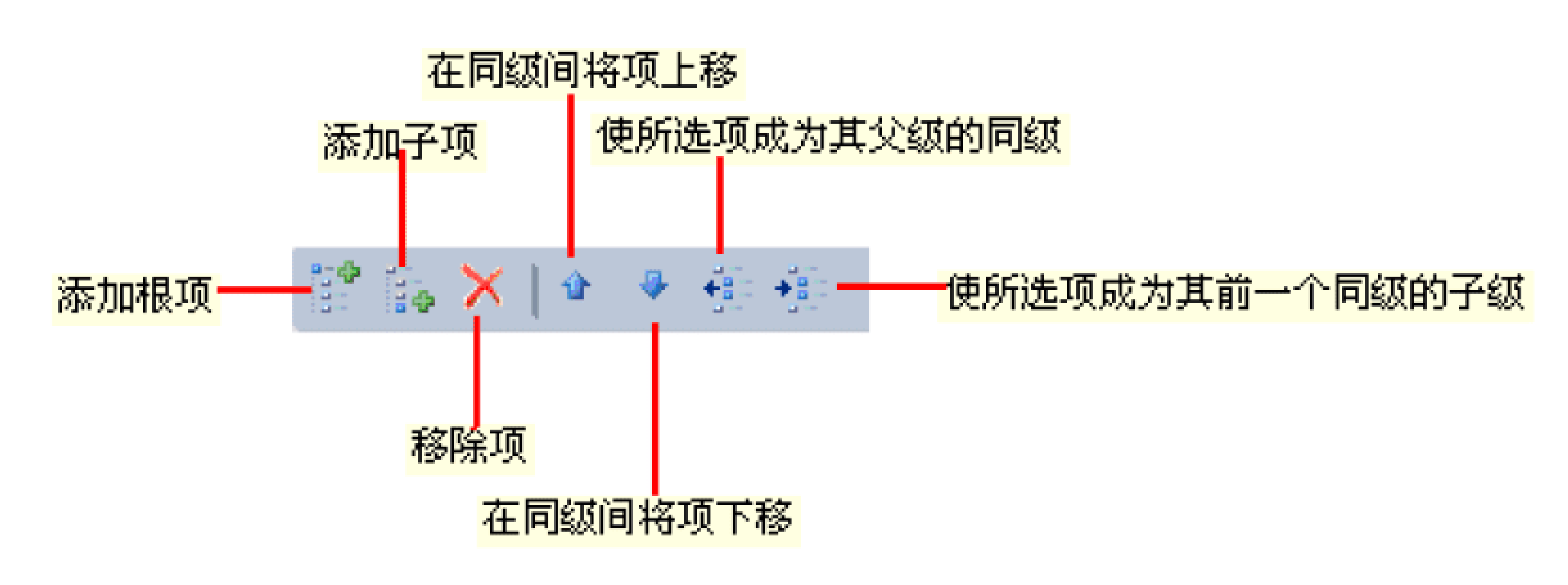


图 3-31　工具栏按钮的功能介绍

(6) 在“菜单项编辑器”窗口右侧的属性选项中，利用 NavigateUrl 属性可以设置各菜单项链接的网页，单击该属性右侧的浏览按钮，打开“选择 URL”对话框，可选择当前网站内的页面，全部设置完成后，单击“确定”按钮，关闭“菜单项编辑器”对话框。

(7) 为了使导航菜单更美观，可以给 Menu 控件应用一下格式，打开“Menu 任务”面板，选择“自动套用格式”命令，然后打开“自动套用格式”对话框，选择“传统型”架构，单击“确定”按钮，如图 3-32 所示。

至此，完成 Menu 控件的设置，切换到 Menu.aspx 的“源”视图，可以看到生成的代码。

(8) 编译并运行程序，在默认浏览器中打开 Menu.aspx 网页，效果如图 3-33 所示。

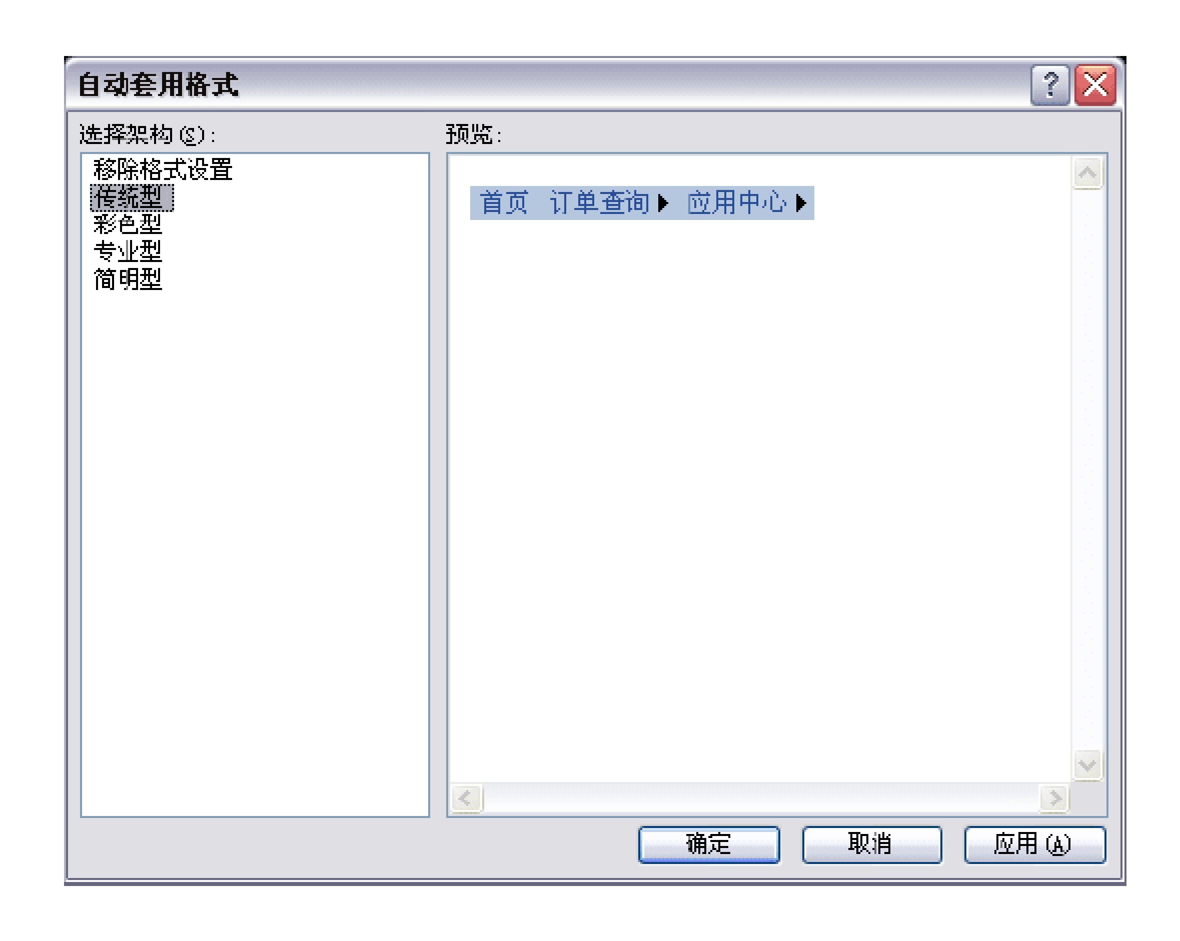


图 3-32　“自动套用格式”对话框

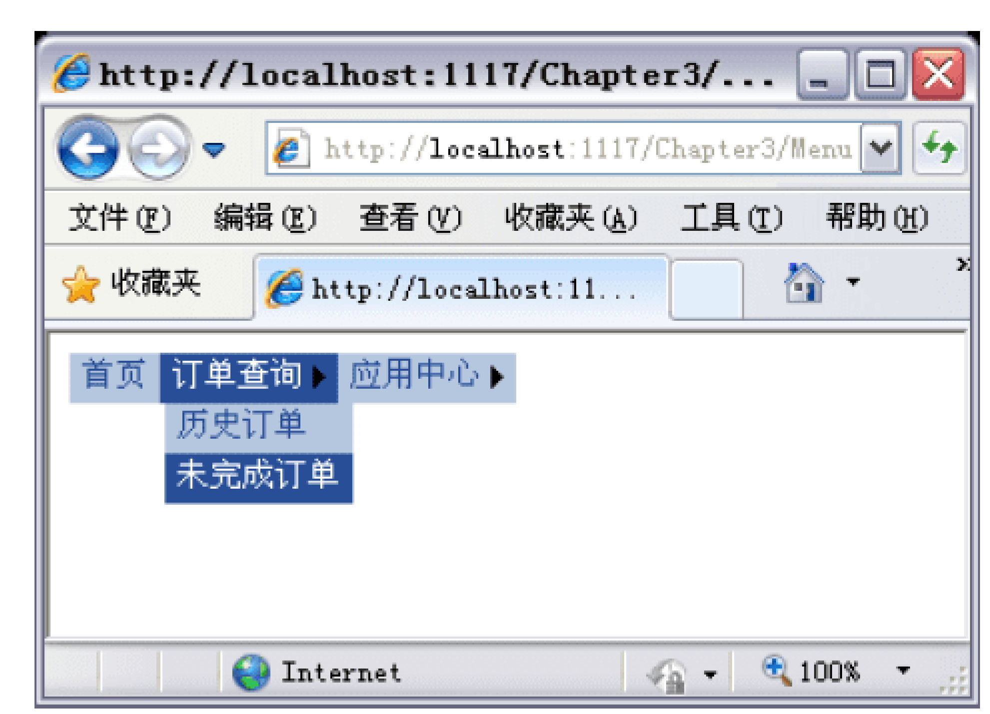


图 3-33　Menu 控件实现的导航效果

3.5.4　使用 TreeView 控件

TreeView 控件与 Menu 控件相似，都提供了导航功能。TreeView 控件与 Menu 控件的区别在于，它不像 Menu 控件那样由菜单项和子菜单组成，而是用一个可折叠的树显示网站的各个部分。根节点下可以包含多个子节点，子节点下又可以包含子节点，最下层是叶子节点。访问者可以快速看到网站的所有部分及位于网站结构层次中的位置。树中的每个节点都显示为一个超链接。

TreeView 控件也包含很多属性，其中常用属性如表 3-10 所示。

表 3-10　TreeView 控件的常用属性

属　　性	说　　明
CollapseImageUrl	节点折叠后显示的图像。默认情况下，常用带方框的+号作为可展开指示图像
CollapseImageToolTip	当用户将鼠标悬停在可折叠菜单项上时显示的工具提示
ExpandImageUrl	节点展开后显示的图像。默认情况下，常用带方框的−号作为可折叠指示图像
EnableClientScript	是否可以在客户端处理节点的展开和折叠事件。默认值为 true
ExpandDepth	第一次显示 TreeView 控件时，树的展开层次数。默认值为 FullyExpand(即−1)，表示全部展开所有节点
ExpandImageToolTip	当用户将鼠标悬停在可展开菜单项上时显示的工具提示
Nodes	设置 TreeView 控件的各级节点及其属性
ShowExpandCollapse	是否显示折叠、展开图像。默认值为 true
ShowLines	是否显示连接子节点和父节点之间的连线。默认值为 false
ShowCheckBoxes	指示在哪些类型节点的文本前显示复选框。共有 5 个属性值：None(所有节点均不显示)、Root(仅在根节点前显示)、Parent(仅在父节点前显示)、Leaf(仅在叶节点前显示)和 All(所有节点前均显示)。默认值为 None
HoverNodeStyle	当鼠标悬停于节点上时，显示节点的样式
LeafNodeStyle	叶节点的样式
LevelStyle	特殊深度节点的样式
NodeStyle	所有节点的默认样式
ParentNodeStyle	父节点的样式
RootNodeStyle	根节点的样式
SelectedNodeStyle	选定节点的样式

例 3-9：利用 TreeView 控件实现网站导航功能。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框，添加名为 TreeView.aspx 的网页。

(2) 打开 TreeView.aspx 文件的“设计”视图，添加 1 个 TreeView 控件，单击 TreeView

控件右上方的小三角，打开“TreeView 任务”面板，在“选择数据源”下拉列表中选择“<新建数据源>”选项，如图 3-34 所示。

(3) 在将打开“数据源配置向导”对话框中选择“站点地图”选项，如图 3-35 所示。

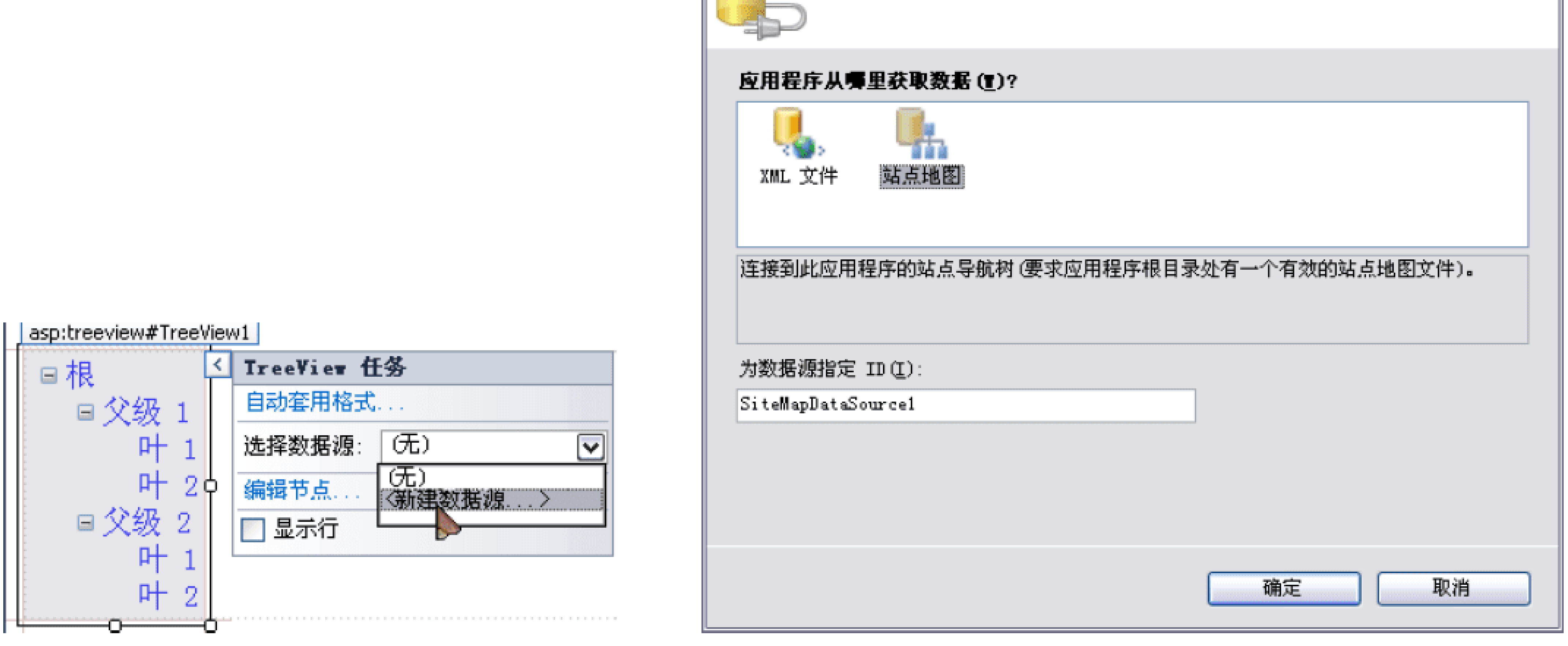


图 3-34　“TreeView 任务”面板　　　图 3-35　“数据源配置向导”对话框

(4) 单击“确定”按钮后，SiteMapDataSource 控件将从 Web.sitemap 文件中得到站点信息，此时的 TreeView 控件如图 3-36 所示。

(5) 编译并运行程序，在浏览器中加载 TreeView.aspx 页面，效果如图 3-37 所示。



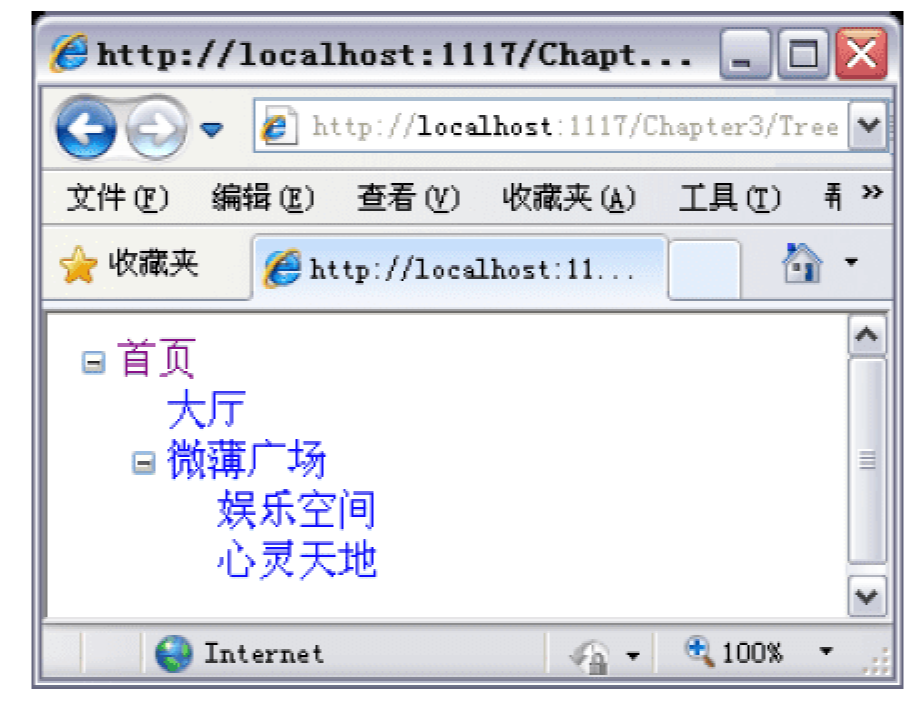


图 3-36　设置数据源后的 TreeView 控件　　　图 3-37　TreeView 导航示例

介绍完导航控件之后，下面给出关于网站导航的一些实用提示，如下所示。

- 当开始构建一个知道将来会增长的 Web 站点时，创建一个逻辑结构。不要把所有文件都放在 Web 站点的根文件夹中，而要将相关的文件根据逻辑组合到同一个文件夹中。这样的逻辑组合可以使站点容易管理，用户容易找到他们要找的页面。虽然用 Web.sitemap 文件很容易将页面移到 Menu 或 TreeView 中，但是如果同时使用了编程方式的重定向或转换时就较难了，因为还需要更新服务器端代码以反映新的站点结构。为了创建稳固的页面结构，可以在开始构建站点之前把它画在纸上，或者使用站点地图图表工具，如 Microsoft Visio。

- 尽量限制显示在 Menu 或 TreeView 控件中的主项和子项的数目。如果供用户选择的选项列表过长，用户会迷路或者混淆。
- 当创建用来存储页面的文件夹时，给它们起简短且有逻辑的名称。

3.6　用户控件

有时可能需要控件具有 ASP.NET 内置服务器控件没有的功能。在这种情况下，用户可以创建自己的控件。有两个选择，可以创建用户控件和自定义控件。

用户控件是能够在其中放置标记和服务器控件的容器。然后，可以将用户控件作为一个单元对待，为其定义属性和方法。

自定义控件是编写的一个类，此类从 Control 或 WebControl 派生。

创建用户控件要比创建自定义控件方便很多，因为可以重用现有的控件。用户控件使创建具有复杂用户界面元素的控件极为方便。

3.6.1　用户控件简介

用户控件可以用来将逻辑上相关的内容和控件组合在一起，然后作为一个单位在内容页、母版页和其他用户控件内使用。

在 ASP.NET 2.0 之前的版本中，用户控件常常用来创建必须出现在站点中每个页面上的可重用的功能块。例如，要创建一个菜单，就会创建一个用户控件，然后将那个控件添加到站点中的每个页面。由于 ASP.NET 对母版页的支持，因此这些情况下不再需要用户控件。这样可以使对站点结构的修改更容易。尽管母版页带来了一些优点，然而在 ASP.NET Web 站点中仍然有利用用户控件的空间。

在某种程度上，用户控件看起来有一些像服务器控件，它们都可以包含能够在页面中重用的编程逻辑和表现。

从开发设计角度来看，用户控件与完整的 ASP.NET 网页(.aspx 文件)非常相似，同时具有用户界面页和代码页。可以采取与创建 ASP.NET 页相似的方式创建用户控件，然后向其中添加所需的标记和子控件。用户控件可以像页面一样包含对其内容进行操作的代码。

但是，用户控件与 ASP.NET 网页也有以下一些区别。

- 用户控件的文件扩展名为.ascx。
- 用户控件中没有@Page 指令，而是包含@Control 指令，该指令对配置及其他属性进行定义。
- 用户控件不能作为独立文件运行。而必须像处理任何控件一样，将它们添加到 ASP.NET 页面中。
- 用户控件中没有 HTML、body 或 form 元素。这些元素必须位于宿主页中。

- 可以在用户控件上使用与在 ASP.NET 网页上所用相同的 HTML 元素(HTML、body 或 form 元素除外)和 Web 控件。例如，如果要创建一个将用作工具栏的用户控件，则可以将一系列 Button 服务器控件放在该控件上，并创建这些按钮的事件处理程序。

3.6.2　创建用户控件

在站点中添加用户控件的方法与添加其他类型的内容相似，只须在“添加新项”对话框中选择“Web 用户控件”选项即可。

一旦向站点中添加了一个用户控件，它就会自动在文档窗口中打开。用户控件没有@ Page 指令，而是有一个@ Control 指令，如下所示：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs"
    Inherits="WebUserControl" %>
```

接下来，就可以使用在前面创建页面时用到的所有工具来设计用户控件。

例 3-10：创建一个实现微调控件的用户控件。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框，添加名为 WebUserControl.ascx 的用户控件。

(2) 切换到 WebUserControl.ascx 的“设计”视图，添加一个 TextBox 控件和两个 Button 控件到设计窗口中。两个 Button 控件的 Text 属性分别设置为 “<” 和 “>”，TextBox 控件的 Text 属性设置为 0，控件布局如图 3-38 所示。

(3) 切换到 WebUserControl1 的代码视图，即打开 WebUserControl.ascx.cs 文件，定义变量，为 Page_Load 事件、两个按钮的单击事件添加如下代码：

```
protected int number;
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        number = int.Parse(TextBox1.Text);
    }
    else
    {
        number = 0;
    }
}
protected void Button2_Click(object sender, EventArgs e)
{
    number++;
    TextBox1.Text = number.ToString();
}
protected void Button1_Click(object sender, EventArgs e)
```

```
        {
            number--;
            TextBox1.Text = number.ToString();
        }
```

(4) 保存用户控件，至此完成用户控件的创建，但还不能在浏览器中查看效果，必须将该用户控件添加到 ASP.NET 页面中才行。



图 3-38　控件布局

3.6.3　使用用户控件

要在 ASP.NET 页面或另一个用户控件中使用一个用户控件，需要执行如下两个步骤。

(1) 注册控件，方法是向希望出现用户控件的页面或控件中添加一个@ Register 指令。

(2) 向页面添加用户控件的标记，并可以(可选地)在其上设置一些特性。

@ Register 指令包含如下 3 个重要的特性。

- src：指向要使用的用户控件。为了在以后的阶段使页面的移动更容易，也可以用~语法指向应用程序根文件夹中的控件。
- tagname：用在页面的控件声明中的标记名。可以自由地命名这个名称，但通常都让它与控件的名称相同。
- tagprefix：容纳用在页面的控件声明中的 TagName 的前缀。正如 ASP.NET 用 asp 前缀指代它的控件一样，也需要为自己的用户控件提供一个前缀。默认情况下，这个前缀是 uc，后跟一个序号，不过也可以将它改为其他内容，如改为公司名称或者自定义的缩略词。

例 3-11：使用例 3-10 中创建的用户控件。

(1) 启动 VWD 2010，打开网站 Chapter3，通过“添加新项”对话框，添加名为 UserControlTest.ascx 的用户控件。

(2) 切换到 UserControlTest.aspx 的“设计”视图，从“解决方案资源管理器”中拖动 WebUserControl.ascx 到页面中。

(3) 切换到“源”视图，可以看到在@Page 指令下方自动生成的@Register 指令，如下：

```
<%@ Register src="WebUserControl.ascx" tagname="WebUserControl" tagprefix="uc1" %>
```

<div>中生成的声明控件的代码如下：

```
<div>
    <uc1:WebUserControl ID="WebUserControl1" runat="server" />
</div>
```

(4) 编译并运行程序，在默认浏览器中打开 UserControlTest.aspx 页面，单击微调按钮即可改变文本框中的数值，如图 3-39 所示。



图 3-39　使用用户控件页面效果

3.6.4　为用户控件添加属性

虽然使用控件创建重复性的内容已经非常有用了，但如果向它们添加自定义逻辑，它们就会更加有用。通过向用户控件添加公有属性或方法，可以影响控件运行时的行为。当向用户控件添加一个属性后，它会自动在正在使用的页面中的控件的“智能提示”中和“属性”面板中变得可用，使得改变外部文件(如页面)中的行为变得更容易。

为了向用户控件中添加属性和方法，可以将它们添加到控件的后台代码文件中。添加的属性可以是各种形式的属性，就像设置标准控件的属性那样。

为了使微调控件更完善，可以为控件添加两个属性：Min 和 Max，分别表示控件所允许的最大和最小值。

例 3-12：为控件添加属性 Min 和 Max。

(1) 启动 VWD 2010，打开网站 Chapter3，打开用户控件 WebUserControl.ascx 的后台代码文件，并添加属性。

为了帮助创建这个属性，VWD 提供了一个简便的代码段。要在 C#中激活该代码段，输入 prop 后按 Tab 键两次。VWD 会添加自动属性的代码结构。但是，这样做会创建一个完整的属性而不是创建一个自动属性，因此在这种情况下，最好手动输入代码。输入代码之后，可以再次按下 Tab 键在字段之间移动，分别输入正确的数据类型和属性名。代码如下：

```
public int Min { get; set; }
public int Max { get; set; }
```

(2) 修改两个按钮控件的单击事件处理程序，当 number 值达到 Min 或 Max 时将不能继续减小或增大，代码如下：

```
protected void Button2_Click(object sender, EventArgs e)
{
    number++;
    if (number > Max)
        number = Max;
```

```
            TextBox1.Text = number.ToString();
            if (number == Max)
                Button2.Enabled = false;
            Button1.Enabled = true;
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            number--;
            if (number < Min)
                number = Min;
            TextBox1.Text = number.ToString();
            if (number == Min)
                Button1.Enabled = false;
            Button2.Enabled = true;
        }
```

(3) 打开 UserControlTest.aspx 页面，选中用户控件，可以在“属性”面板中设置新添加的属性 Min 和 Max，如图 3-40 所示。

(4) 如果在“源”视图中直接输入代码，也可以从智能提示窗口中看到这两个属性，如图 3-41 所示。

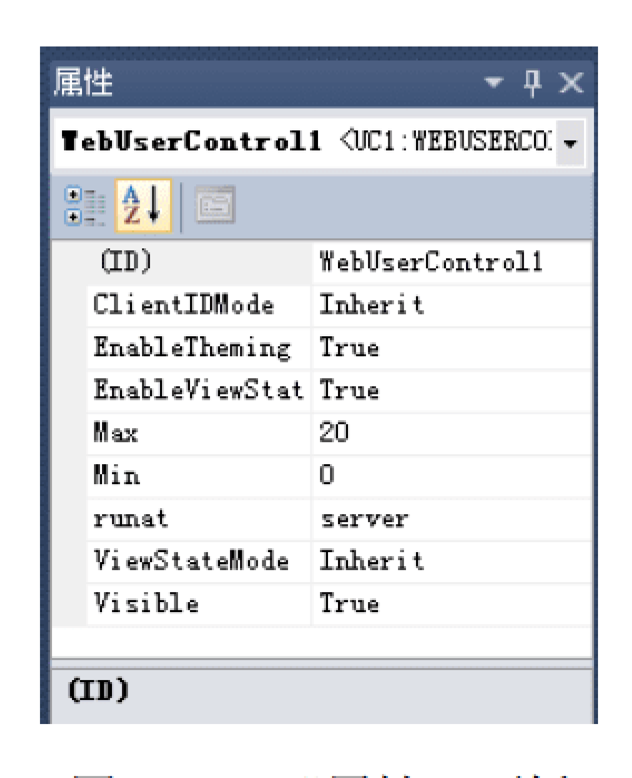


图 3-40　“属性”面板

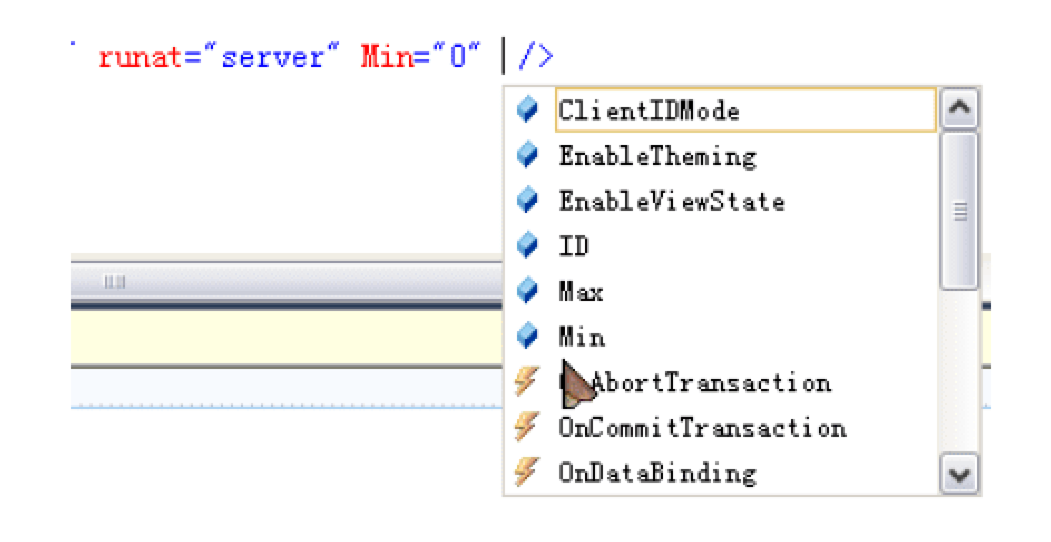


图 3-41　智能提示窗口

(5) 设置了 Min 和 Max 属性后，当文本框中的值达到最大值时，将不能再增大。当达到最小值时，将不能再减小。

3.6.5　用户控件的站点范围注册

如果用户控件要在多个页面中使用，则可以在 web.config 文件中全局地注册这个控件。这样，它就会变得在整个站点内可用，而不需要在每个页面上注册。

例 3-13：在 web.config 文件中注册用户控件。

(1) 打开站点根文件夹中的 web.config 文件。

(2) 在<system.web>元素内添加如下代码，其中包含一个带有<add />子元素的<controls />元素：

```
<pages theme="Monochrome">
  <controls>
    <add tagPrefix="uc1" tagName="WebUserControl" src="~/WebUserControl.ascx" />
  </controls>
</pages>
```

(3) 保存修改并关闭文件。

(4) 此时，在 ASP.NET 网页中添加用户控件时，就不需要@Register 指令了。

提示：

用户控件对于封装重复内容非常有效，但是它们也会使站点难以管理，因为代码和逻辑包含在多个文件中。所以，不要过度使用用户控件。如果不确定有些内容是否会在站点的另一部分中重用，可以先直接把它嵌在页面中。如果有需要，再将它移到单独的用户控件中即可。

3.7　本章小结

本章介绍了 ASP.NET 服务器控件以及用户控件。使用 ASP.NET 服务器控件，可以大幅减少开发 Web 应用程序所需编写的代码量，提高开发效率和 Web 应用程序的性能。本章重点介绍了标准控件、HTML 控件、验证控件和导航控件，其他类别的控件将在其他章节介绍或不作为本书所学内容。

在已有控件的基础上，还可以开发更实用的用户控件，用户控件能够大大提高站点的可维护性。不需要在站点中的很多不同页面上重复相同的标记和代码，可以将代码封装到单个控件中，然后在站点的不同区域使用该控件。

3.8　思考和练习

1. TextBox 控件的 TextMode 属性有哪几个可选值，各是什么含义。

2. 如何将 RadioButton 控件进行分组？

3. ASP.NET 提供了几个验证控件，各有什么作用？

4. 当使用一个 CustomValidator 控件时，可以在客户端和服务器上编写有效性验证代码。如何告知 ASP.NET 运行库在有效性验证处理期间调用什么客户端有效性验证方法？

5. 使用 TreeView 控件有两种方式：一种方式是作为带项和子项的列表，单击它们时能折叠或展开；另一种方式是作为显示所有项的静态列表，不能折叠或展开。要禁止用户展开或折叠树中的项，需要设置控件上的什么属性呢？

6. 注册用户控件需要使用什么指令？

7. 如何为用户控件添加属性？

第4章　页面设计与布局

开发 Web 应用程序通常需要考虑两个方面，即功能和外观。ASP.NET 提供了一些可在应用程序中对页面、控件的外观和样式进行自定义的功能，例如可以为某个控件设置字体、背景色和前景色、宽度以及高度等样式。本章将全面研究 Web 应用程序中样式控制和页面布局所用到的技术和使用方法，包括 CSS 样式、主题和母版页。这些技术对于创建具有一致外观的网站非常有用，也有利于使站点看起来更有吸引力，也更加专业。

本章学习目标：

- 编写和应用 CSS 样式
- VWD 提供的大量快速编写 CSS 的工具
- 创建和应用主题
- 在主题中定义外观
- skinID 属性的使用
- 创建母版页和内容页

4.1　CSS 样式

Internet 出现伊始，Web 页面主要由文本和图像组成。文本是使用纯 HTML 格式化的，这种格式化所提供的样式化页面的选项很有限。因此诞生了 CSS 来弥补这方面的缺陷。

本节将介绍 CSS 的概念、在 VWD 2010 中使用 CSS 以及 CSS 的样式规则和用法。

4.1.1　HTML 格式化的不足

使用 HTML 进行格式化的问题之一是它提供的样式化页面的选项很有限。可以用<em>、<strong>及<font>这样的标记来改变文本的外观，用 bgcolor 这样的属性来改变 HTML 元素的背景颜色，还有几个属性可用来改变链接出现在页面中的方式。但是很显然，这个功能集不足以创建符合用户期望与需求的生动 Web 页面。

另一方面，在设计时，HTML 会强制要求在 HTML 文档中嵌入格式化信息，使得以后难以对设计进行重用或修改。

除了维护性问题之外，HTML 格式化的另一个问题是导致用户的浏览器中不能轻松地在运行时修改格式。

HTML 格式化的最后一个问题是，页面中的附加标记大大增加了页面的大小。这样，由于需要从 Web 站点中的各个页面上下载信息，下载和显示就会变慢。而且，当需要滚动

大型的 HTML 文件来查找需要的内容时，页面也会变得难以维护。

简言之，使用 HTML 格式化存在如下一些问题。

- 它的有限功能集远远满足不了页面的格式化需求。
- 数据与表现混合在相同的文件中。
- HTML 无法在浏览器中于运行时轻松地切换格式。
- 必需的格式化标记与属性使页面更大，因此加载和显示更慢。

4.1.2　CSS 简介

CSS(Cascading Style Sheet)，中文译为层叠样式表，是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。就语法而言，CSS 是一种容易学习的语言。它的“语法”仅由几个概念组成，使得它相当容易入门。CSS 的难点在于所有主流浏览器呈现页面的方式。尽管实际上每种浏览器都能够理解 CSS，但当根据 CSS 标准显示页面时，它们都有各自的“怪癖”。这个标准是由提出 HTML 标准的同一个组织 W3C(World Wide Web Consortium)提出的，它出现过 3 个版本：1.0、2.1 和 3.0。在这 3 个版本中，2.1 版本是现在人们最普遍接受的，它包含了版本 1.0 中的所有内容，但是在其基础上又增加了大量功能。这个版本也是 VWD 默认使用和生成的版本。

CSS 几乎能在所有可能的方面格式化 Web 页面。它提供了一套丰富的选项，可以修改 Web 页面的各个细微方面，包括字体(大小、颜色和字体等)、颜色和背景色、围绕 HTML 元素的边框、元素在页面中的位置以及其他很多方面。当今所有的主流浏览器基本都能接受 CSS，它就是 Web 页面可视化表现的语言，而且在 Web 开发人员中非常流行。

CSS 规定了两种定义样式的方法，即内联式和级联式。

1. 内联式样式

直接将样式控制放在单个 HTML 元素内，称为内联式或行内样式。该样式通过 style 属性来控制每个元素的外观，此方法直观但是很繁琐。除非具有相同样式的元素较少，否则很少采用。下面是一段采用内联式来控制各个元素外观的示例代码：

```
<body style="text-align:center">
<form id="form1" runat="server">
<div style="text-align:center; width:400px; border:solid 1px blue">
<h1 style="font-size:x-large; color:red ">欢迎光临</h1>
<h2 style="font-size:large; color:blue ">这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

2. 级联式样式

在网页的 head 部分定义或导入的样式，称为级联式样式。该样式可以实现将网页结构和表现分离，这样，当修改某些元素的样式时，只需要修改 head 部分定义或引入的样式，

该网页内所有具有相同样式的元素都会自动应用新的样式。

级联式样式又可以分为两种方式，即内嵌式和链接式。

● 内嵌式

在 head 部分直接实现的 CSS 样式，称为内嵌式。这种 CSS 一般位于 HTML 文件的头部，即在<head>与</head>标签内，并且以<style>开始，以</style>结束。例如，将上面示例代码中的样式抽取出来，以内嵌式级联样式修改将得到如下代码：

```
<head>
 <title>内嵌式样式</title>
<style Type="text/css">
<!--
body{ text-align:center }
div{ text-align:center; width:400px; border:solid 1px blue }
h1{ font-size:x-large; color:red}
h2{ font-size:large;   color:blue }
-->
</style>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1>欢迎光临</h1>
<h2>这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

其中<style>与</style>之间是样式的内容，在{ }前面可以写样式的类型和名称。{ }中是样式的属性。这种方法是经常被使用的添加样式表的方法。

可见，采用内嵌式比内联式方便了很多，body 内的代码也相对简洁，修改某个元素的样式时只需修改 head 内的代码即可。

● 链接式

内嵌式只解决了某一个网页内部结构和表现分离的问题，但通常网站都由很多网页组成，不同网页中的某些元素采用了相同的样式，仍然需要分别设置，因此，将样式放在一个单独的 CSS 文件中，然后通过为每个网页引入该文件来实现统一的外观将是一种更好的选择。

在 head 部分通过导入以扩展名为.css 的文件来实现 CSS 样式，称为链接式。利用这种方法在网页中可以调用已经定义好的样式表来实现样式表的应用，定义好的样式表通常单独以文件的形式存放在站点目录中。这种方法实现了将网页结构和表现的彻底分离，最适合大型网站的 CSS 样式定义。

例如，将上面示例中的样式抽取出来以文件的形式存放，保存到一个名为 style.css 的

文件中，内容如下：

```
body
{
    text-align:center;
}
div
{
    text-align:center;
    width:400px;
    border:solid 1px blue;
}
h1
{
    font-size:x-large;
    color:red;
}
h2
{
    font-size:large;
    color:blue;
}
```

在页面中，可以通过<link>标记引用样式文件 style.css，代码如下所示：

```
<head>
    <title>链接式样式</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h1>欢迎光临</h1>
            <h2>这是一个被 style 修饰的页面</h2>
        </div>
    </form>
</body>
```

在引用样式的标记<link>中，ref 属性规定了 XHTML 与被链接文件的关系，href 属性指定了要链接的样式表文件的 URL，type 属性则规定了链接文件的类型。上述样式文件是和当前页面在同一个目录下存放，如果不在同一个目录下存放，相应的<link href=" "…>标记中 href 属性的值则会有所改变。

3. 样式嵌套

此外，如果某个元素既引用了链接样式文件中定义的样式，又在 head 部分定义了新的

样式，或者在元素内部通过 style 属性定义了新的样式，那么该标记元素最终呈现的效果会是什么样呢？下面通过一个例子来说明这个问题。

例 4-1：样式嵌套举例。

```
<head>
<title>链接式样式</title>
<style Type="text/css">
<!--
h1 { font-weight:bold }
    h2 { color: yellow}
-->
</style>
<link rel＝"stylesheet" href="style.css" type="text/css">
</head>
<body>
<form id="form1" runat="server">
<div>
<h1 style=" font-size:small ">欢迎光临</h1>
<h2 style=" font-weight:bold ">这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

其中，style.css 文件是前面创建的样式文件。运行这个 HMTL 文件，在浏览器中可以看到，h1 元素内的文字以粗体、小号、红色显示，而 h2 元素内的文字则以粗体、大号、蓝色显示。可见，链接式样式中 h1 元素的 font-size 属性和内嵌式样式中 h2 元素的 color 属性都没有起作用，而不冲突的样式则都会起作用。这就是样式嵌套中的冲突问题，浏览器解决这种问题的方法就是一旦发现样式冲突，则通过“就近使用”原则，采用距离该元素最近的样式进行显示，而不冲突的样式则通过顺序组合后形成最终样式进行显示。

一般情况下，在样式表(.css)文件中定义适合大多数网页公用的样式，在网页内部采用内嵌式定义该页面特有的样式，内联式样式定义个别元素的样式，再结合可视化的开发工具，从而使样式控制真正灵活、方便。

4.1.3　CSS 属性

属性是元素的一部分，可通过样式表修改。CSS 规范定义了一个长属性列表，但在大多数 Web 站点中不会用到所有项。表 4-1 列出了部分常见的 CSS 属性，并说明了它们的应用场合。

表 4-1　常见的 CSS 属性

CSS 属性	描　述	示　例
background-color background-image	指定元素的背景色或图像	background-color:White; background-image: url(Image.jpg);
border	指定元素的边框	border: 3px solid black;
color	修改字体颜色	color: Green;
display	修改元素的显示方式，允许隐藏或显示它们	display: none; 这种设置使元素被隐藏，不占用任何屏幕空间
float	允许用左浮动或右浮动将元素浮动在页面上。其他的内容则被放在相对的位置上	float: left; 该设定使跟着一个浮动的其他内容被放在元素的右上角。本章也将介绍它的工作原理
font-family font-size font-style font-weight	修改页面上使用的字体外观	font-family: Arial; font-size: 18px; font-style: italic; font-weight: bold;
height width	设置页面中元素的高度或宽度	height: 100px; width: 200px;
margin padding	设置元素内部(内边距)或外部(页边距)的可用空间	padding: 0; margin: 20px;
visibility	控制页面中的元素是否可见。不可见的元素仍然会占用屏幕空间，只是看不到它们而已	visibility: hidden; 这会使元素不可见。但仍然会占用页面的原始空间

VWD 会通过它的许多 CSS 工具找到恰当的属性，因此不必全部记住它们。

4.2　在 VWD 中使用 CSS

VWD 中有几个使用 CSS 的便利工具，如下所示。

- “样式表”工具栏：用来快速访问并创建新规则与样式。
- “CSS 属性”面板：用来修改属性值。
- “管理样式”窗口：用来组织站点的样式，将它们从内嵌样式表改为外部样式表，反之亦然；对它们重新排序；将现有样式表链接到一个文档；创建新的内联、内嵌或外部样式表。

- “应用样式”窗口：用来从站点中选择所有可用样式，并将它们快速应用到页面中的不同元素上。
- “添加样式规则”对话框：用于构建较复杂的选择器。

4.2.1　新建样式

在 VWD 2010 中使用 CSS 非常方便，即可以在“源”视图中利用 VWD 的“智能提示”功能设置各种样式，也可以利用可视化的对话框和便利工具快速完成各种样式的设置。

1. 在源视图下设置样式

在“源”视图下，利用系统提供的智能提示功能，可以方便地设置各种元素的样式内联式样式，具体步骤如下。

(1) 在想要设置格式的 HTML 标记内，输入 style="，并按空格键，将弹出 VWD 2010 提供的“智能提示”工具，如图 4-1 所示。

(2) 定义任意数量的属性(“属性:值”对)，属性之间用分号分隔。

2. 在可视化窗口中设置样式

利用可视化窗口设置样式的方法有很多，可以在“源”视图或者“设计”视图下选中某个标记元素，然后单击“属性”面板中 style 属性后面的省略号按钮，将打开“修改样式”对话框，如图 4-2 所示。

该对话框分为两个窗格，左窗格列出了 9 个类别，当选择某个类别时，右窗格中将显示所选类别下的选项。设置了样式选项并单击“确定”按钮后，新的样式定义将自动在“源”视图中生成，也可以在“设计”视图下查看最新的效果。

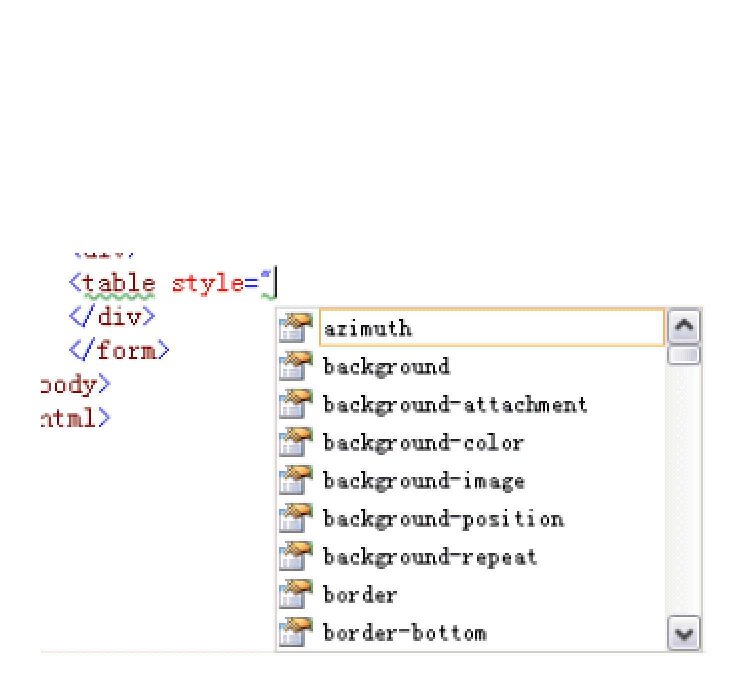


图 4-1　VWD 的“智能提示”工具



图 4-2　“修改样式”对话框

这种方法只能将定义的样式属性以内联式生成，放在每个元素的 style 属性中。如果要定义内嵌式样式可以使用如下步骤。

(1) 切换到“设计”视图，在“格式设置”工具栏的“目标规则”列表中，选择“应用新样式”选项，如图 4-3 所示。

(2) 此时将打开“新建样式”对话框，该对话框与“修改样式”对话框相似，所不同

的是“新建样式”对话框中包含了“选择器”用于选择对哪一个标记进行定义，以及通过“定义位置”将当前定义存放到哪里，如图 4-4 所示。

在“选择器”列表中选择某个选择器，如 bgsound，就可以创建应用于所有 bgsound 元素的样式。“定义范围”列表设置为“当前网页”，表示该样式规则在当前页的 style 元素中创建。若想查看已创建的样式规则，可以切换到“源”视图并滚动到 style 元素，该元素位于<head>标记内。

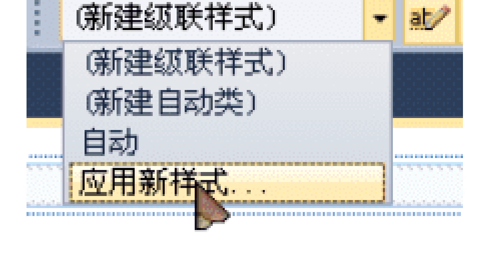


图 4-3　选择应用新样式



图 4-4　“新建样式”对话框

3. 使用“CSS 属性”面板

可以使用“CSS 属性”面板对正在定义的内嵌式样式规则进行修改，具体步骤如下。

(1) 单击要修改样式的元素，选中部分以蓝色框包围并有一个标签指示 h3 元素已选中。

(2) 选择“视图”|“CSS 属性”命令，打开“CSS 属性”面板，可以看到 h3 元素的 CSS 属性列表，如图 4-5 所示。

(3) 通过该面板可以根据需要设置 CSS 属性。

在 CSS 属性列表中所做的修改可以立即通过“设计”视图显示出来，如果要观察样式代码，可以切换到“源”视图并滚动到 style 元素处进行查看。

4. 新建样式表文件

使用 CSS 的另一个有效方法是将样式规则放入独立的样式表文件中。然后所有页面都可以引用这些样式，这样可以使这些页面看起来非常一致。

例 4-2：新建样式表文件，并将例 4-1 中 h2 元素的样式副本添加到新建的样式表文件中。

(1) 启动 VWD 2010，选择“文件”|“新建网站”命令，新建空网站 Chapter4。

(2) 在“解决方案资源管理器”中，右击解决方案的名称，从弹出的快捷菜单中选择“添加现有项”命令。将例 4-1 中的 HTML 文件添加到项目中。

(3) 通过“添加新项”对话框，在“模板”列表中选择“样式表”选项，新建样式表 StyleSheet.css。此时，编辑器将打开一个包含空 body 样式规则的新样式表。

(4) 选择“视图”|“管理样式”命令，打开“管理样式”窗口，此时该窗口中会列出样式文件中的样式和当前网页中的内嵌样式。

(5) 右击当前网页中的 h2 样式，从弹出的快捷菜单中选择“新建样式副本”命令，如图 4-6 所示。



图 4-5　“CSS 属性”面板

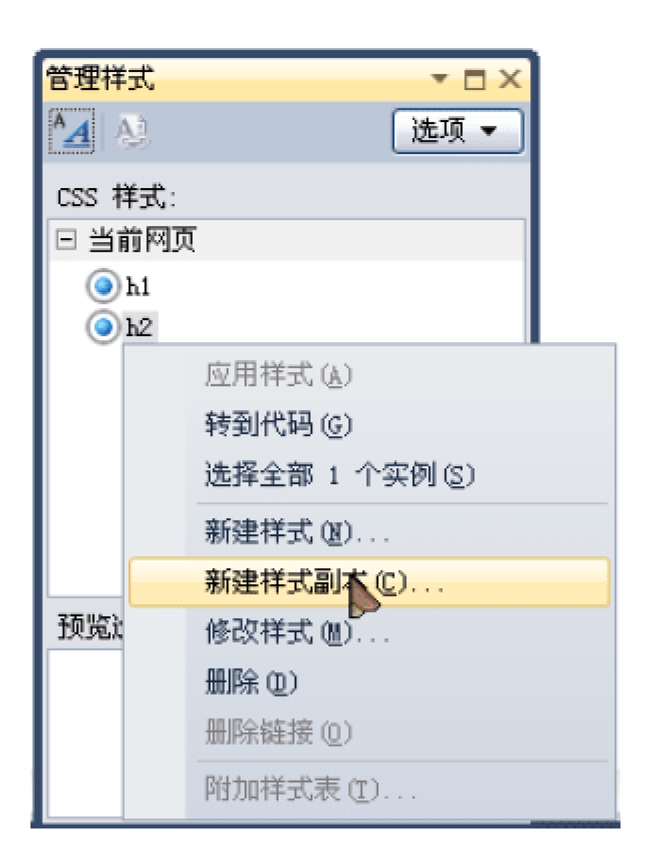


图 4-6　“管理样式”窗口

此时将打开“新建样式”对话框，允许创建一个基于当前选项的新样式，在“选择器”下拉列表中将默认自动选择 h2，在“定义位置”下拉列表中选择“现有样式表”选项，在 URL 下拉列表中选择 StyleSheet.css 选项，如图 4-7 所示。



图 4-7　“新建样式”对话框

(6) 单击“确定”按钮关闭对话框。VWD 会创建 h2 样式的一个副本，并把它放在样式表文件 StylesSheetcss 中。

接下来，可以将当前网页中的 h2 样式删除，改为使用样式表文件中的样式。

(7) 在“管理样式”窗口中，再次右击当前网页中的 h2 样式，从弹出的快捷菜单中选择“删除”命令。这样会把样式属性从<head>标记内删除。

(8) 打开或切换到例 4-1 中的 HTML 页，切换到“设计”视图。选择“格式”|“附加样式表”命令，打开“选择样式表”对话框，选择刚才创建的样式表文件 StyleSheet.css，单击“确定”按钮。

(9) 执行上述操作后，在“源”视图中可以看到，在<head>标记中将添加如下代码：

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

通过以上步骤即可完成将内嵌式样式移到样式表文件中的操作。

4.2.2　样式规则

一个样式表由若干个样式规则组成。样式规则是指网页元素的样式定义，包括元素的显示方式以及元素在页中的位置等。打开前面添加的样式表文件 StyleSheet.css，在其中单击鼠标右键，从弹出的快捷菜单中选择“添加样式规则”命令，将打开如图 4-8 所示的“添加样式规则”对话框。

在该对话框中选择某个元素，也可定义一个类或定义一个元素 ID，单击“确定”按钮即可添加一个新的样式规则。例如，添加一个元素 img，在样式表文件中可以看到如下新建的样式规则。

```
img
{
}
```

该规则默认是仅有元素名称的空规则，在大括号内单击鼠标右键，从弹出的快捷菜单中选择“生成样式”命令，即可打开“修改样式”对话框，进行样式定义和修改。

无论是定义内嵌式样式还是链接式样式，每个样式的定义格式都是一样的，如下所示：

```
样式定义选择符{ 属性 1:值 1;  属性 2:值 2;   ……   }
```

其中，样式定义选择符是指样式定义的对象，可以是 HTML 标记元素，也可以是用户自定义的类、用户自定义的 ID、伪类和伪元素等。

1. 标记选择符

任何 HTML 元素都可以是一个 CSS 的标记选择符。标记选择符仅仅是指向特别样式的元素。“添加样式规则”对话框中的“元素”下拉列表中提供了所有可供使用的标记选择符。

2. 类选择符

每一个标记选择符都能自定义不同的类，从而允许同一元素具有不同的样式。指定某个标记选择符内的自定义类的一般形式为：

```
标记选择符.类名{样式属性 1:值 1;  样式属性 2:值 2;  ……}
```

在“添加样式规则”对话框中先选择“类名”单选按钮，在文本框中输入 one，然后选中“可选元素”复选框，在其下拉列表中选择 p 元素，即可自动生成 p.one 样式规则，如图 4-9 所示。



图 4-8　“添加样式规则”对话框

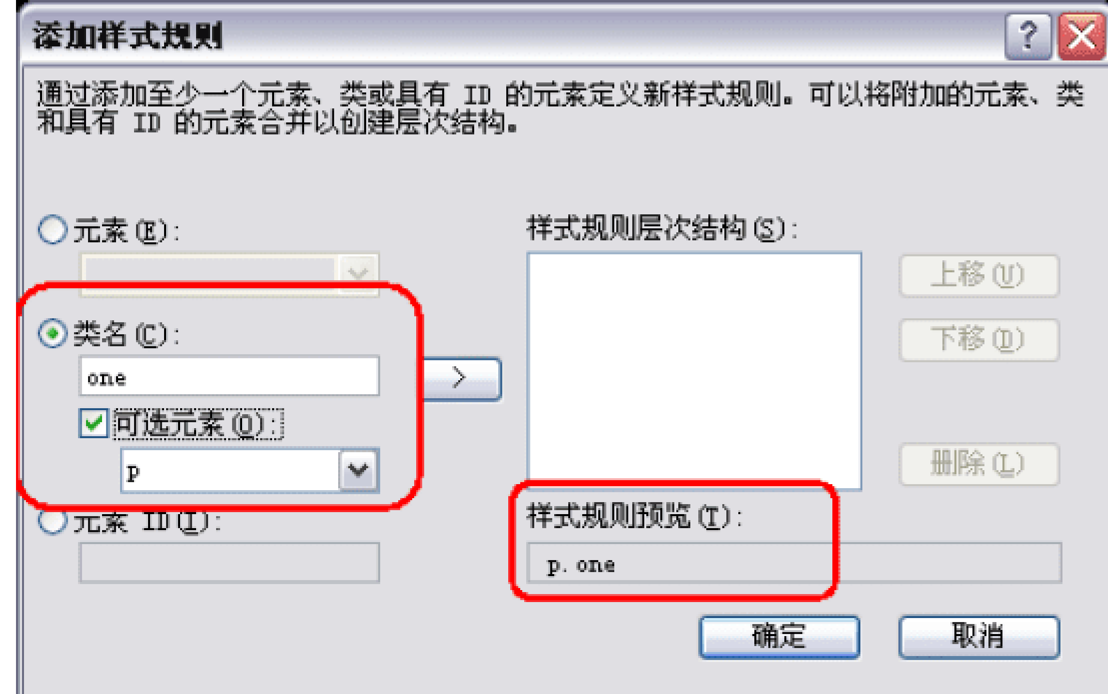


图 4-9　添加 p.one 样式规则

单击“确定”按钮将生成仅有类名称的空规则，在大括号内单击鼠标右键，从弹出的快捷菜单中选择“生成样式”命令，打开“修改样式”对话框，设置字体的 color 属性为红色，相应的代码如下：

```
p.one
{
 color: red;
}
```

用同样的方法，定义 p.two 类选择符样式，设置字体的 color 属性为蓝色。

接下来，就可以在代码中引用类选择符，其方法是通过元素的 class 属性来实现的，例如，下面的代码，其显示效果如图 4-10 所示。

```
<p class="one">类别选择器 1</p>
<p class="two">类别选择器 2</p>
```

其含义是在 p 中引用 one 会以红色样式显示，在 p 中引用 two 会以蓝色样式显示。

类选择符的定义也可以与标记选择符无关，这样，类选择符就可以应用于任何元素。这种自定义类选择符的形式如下：

```
.类名{样式属性 1:值 1; 样式属性 2:值 2; ……}
```

创建这种类选择符时，只要不选中“可选元素”复选框即可。

3. ID 选择符

ID 选择符用于分别定义每个具体元素的样式。一个 ID 选择符的指定要有指示符#在名字前面。使用时通过指定元素的 id 属性来关联。例如：

```
#index { color:blue }
```

引用时，使用 id 属性声明即可。

```
<p id="index">本段落的颜色为蓝色</p>
```

自定义 ID 选择符与自定义类选择符的方式非常相似，在“添加样式规则”对话框中，

选择“元素 ID”单选按钮，输入相应的名称即可。但是两者在使用上是有区别的。在同一个网页中，多个标记元素可以使用同一个自定义类选择符，而 ID 选择符只能为某一个标记元素使用。这种选择符应该尽量少用，因为它有一定的局限性。

技巧：

如果在一个元素的样式定义中，既有标记选择符，又有自定义类选择符和自定义 ID 选择符，那么自定义 ID 选择符的优先级最高，其次是自定义类，标记选择符的优先级最低。

4. 关联选择符

关联选择符是一个用空格隔开的两个或更多的单一标记选择符组成的字符串。一般格式如下：

```
选择符 1　选择符 2　……　{属性:值; ……}
```

这些选择符具有层次关系，并且它们的优先级比单一的标记选择符大。例如：

```
p h2{ color:red }
```

这种定义方式只对 p 元素所包含的 h2 元素起作用，单独的 p 或者单独的 h2 元素均无法应用该样式。在“添加样式规则”对话框中先添加 p 选择符，单击“>”按钮将其添加到“样式规则层次结构”中，然后在添加 h2 元素，如图 4-11 所示，如果层次结构有变化，还可以通过“上移”和“下移”按钮进行修改。

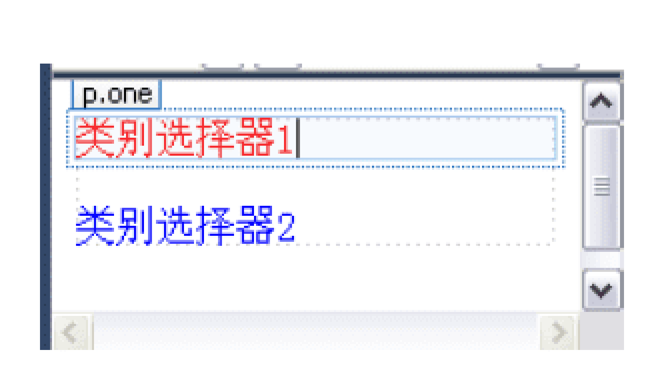


图 4-10　引用类选择符应用样式



图 4-11　定义关联选择符

这种方式不仅适用于标记选择符，还可以关联自定义用户类，自定义 ID 以及任何样式选择符。

5. 并列选择符

如果有多个不同的元素定义的样式相同，则可以使用并列选择符简化定义。例如：

```
h1,h2,h3{ color:blue}
```

每个元素之间用逗号隔开，表示所有的 h1、h2、h3 标记中的内容都将以蓝色样式显示。

6. 伪类

伪类是 CSS 中非常特殊的类，它能自动地被支持 CSS 的浏览器所识别。伪类可以指

定 XHTML 中的<a>元素以不同的方式显示链接(links)、已访问链接(visited links)和可激活链接(active links)。其中，一个已访问链接可以定义为不同颜色的显示，甚至不同字体大小和风格。

CSS 中用 4 个伪类来定义链接的样式，分别是：a:link、a:visited、a:hover 和 a:active，例如：

```
a:link{font-weight : bold ;text-decoration : none ;color : #C00000 ;}
a:visited {font-weight : bold ;text-decoration : none ;color : #C30000 ;}
a:hover {font-weight : bold ;text-decoration : underline ;color : #F60000 ;}
a:active {font-weight : bold ;text-decoration : none ;color : #F90000 ;}
```

以上语句分别定义了链接、已访问过的链接、鼠标停在上方时、点下鼠标时的样式。注意，必须按以上顺序写，否则显示可能和预想的不一样。

4.2.3　应用样式

在 VWD 中，通过“应用样式”对话框可以快速地为页面中的元素应用已经定义好的 CSS 样式。选择“视图”|“应用样式”命令即可打开“应用样式”窗口。

例 4-3：通过“应用样式”窗口对页面中的元素应用样式规则。

(1) 首先，在要应用样式的页面的“设计”视图，选择“格式”|“附加样式表”命令，将前面创建的样式表文件 StyleSheet.css 附加到页面。

(2) 选择“视图”|“应用样式”命令，打开“应用样式”窗口。

(3) 该窗口显示了它在当前页面中找到的所有选择器和附加的任何样式表。如果没有看到，也可以单击该窗口工具栏中的“选项”按钮并选择“显示所有样式”命令。

(4) 将光标定位到要应用样式的元素处，如<p>元素，此时“应用样式”窗口中将显示当前可用的样式，单击 p.one 类，或者单击右侧的下拉菜单按钮，从弹出的下拉菜单中选择“应用样式”命令，如图 4-12 所示。



图 4-12　应用样式

(5) 切换到“源”视图，可以看到 VWD 会向<p>标记添加一个 class 属性：

```
<p class="one" >CSS 样式示例</p>
```

说明：

如果要应用多个类，则在单击列表中其他类时，按住 Ctrl 键，这样就会应用一个类列表，元素的 class 属性之间由一个空格隔开。

(6) 单击“应用样式”窗口中的“清除样式”选项，可以快速地从标记中删除现有类和内联样式。

4.3　页面布局

除了为页面的内容设置样式，页面元素的布局和定位是否合理也是衡量网页设计是否美观的重要指标。本节将介绍网页的基本布局方式、页面元素的定位以及表格布局和层布局。

4.3.1　网页的基本布局方式

常见的网页布局方式有左对齐、居中和满宽度显示。默认情况下，网页内容是水平左对齐的，然而，在实际页面中，常用的布局方式是页面水平居中和满宽度显示。

1. 页面水平居中

设置页面水平居中的方法是在 body 的 style 样式中设置 text-align 属性的值为 center。如果还希望页面的宽度固定，则可以通过设置 div 的 width 属性来实现。例如下面的代码：

```
<body style="text-align:center; ">
  <form id="form1" runat="server">
    <div id="div1" style="width:760px; text-align:center; height:200px"></div>
  </form>
</body>
```

2. 页面满宽度显示

设置页面满宽显示的方法是将 div 的固定宽度设置为百分比，这样宽度就会随显示界面的大小自动调整。例如下面的代码：

```
<body style="text-align:center; ">
  <form id="form1" runat="server">
    <div id="div1" style="width:98%; text-align:center; height:200px"></div>
  </form>
</body>
```

这种布局方式的优点是，无论浏览器是否最大化显示，都不会出现横向滚动条；缺点是页面元素相对位置不固定，不利于用户和窗体之间的操作。

4.3.2 页面元素定位

页面元素的定位分为流布局和坐标定位布局两种，其中，坐标定位布局又分为绝对定位和相对定位，这里仅介绍流布局和坐标绝对定位。

1. 流布局 static

如果采用该布局，则页面中的元素将按照从左到右、从上到下的顺序显示，各元素之间不能重叠。如果不设置元素的定位方式，则默认就是流式布局。

2. 坐标绝对定位 absolute

在使用坐标绝对定位之前，必须先将 style 元素的 position 属性设置为 absolute，然后就可以由 style 元素的 left、top、right、bottom 和 z-index 属性来决定元素在页面中的绝对位置。left 属性表示元素的 x 坐标，top 属性表示元素的 y 坐标，坐标的位置是以它最近的具有 position 属性的父容器为参照物的。

例 4-4：页面元素的定位方式演示。

(1) 启动 VWD 2010，打开网站 Chapter4，通过“添加新项”对话框在该网站中添加一个名为 Default.aspx 的 Web 窗体页。

(2) 修改页面<body>内的代码如下所示：

```
<body>
    <form id="form1" runat="server">
        <div id="div1" style="border: 1px #000080 solid; text-align: left; width:350px; height: 200px;">
            <div id="div2" style="width: 200px; height: 120px; text-align: left; border: 1px #00FF00 solid;
background-color: #E080F0">
                <div id="div3"style="position: absolute; top: 76px; left: 123px; width: 150px; height:
100px; border: 2px #800000 solid; background-color: #FFFF00">
                    <div id="div4" style="position: absolute; left: 23px; top: 30px; width: 100px; height:
60px; border: 3px #FF00FF solid; background-color: #00FFFF">
                    </div>
                </div>
            </div>
        </div>
    </form>
</body>
```

然后切换到“设计”视图，观察其显示的效果，如图 4-13 所示。运行该页面，可以看到，无论浏览器窗口如何变化，各层之间的位置仍然保持不变。

(3) 具有不相同 z-index 值的元素可以重叠，其效果就像多张透明的纸按顺序叠放在一起。其中，z-index 值大的元素会覆盖 z-index 值小的元素。为 div3 元素增加 z-index 属性，

如下：

```
<div id="div3" style="position: absolute;   ……   z-index: -1;">
```

此时，“设计”视图的效果如图 4-14 所示。

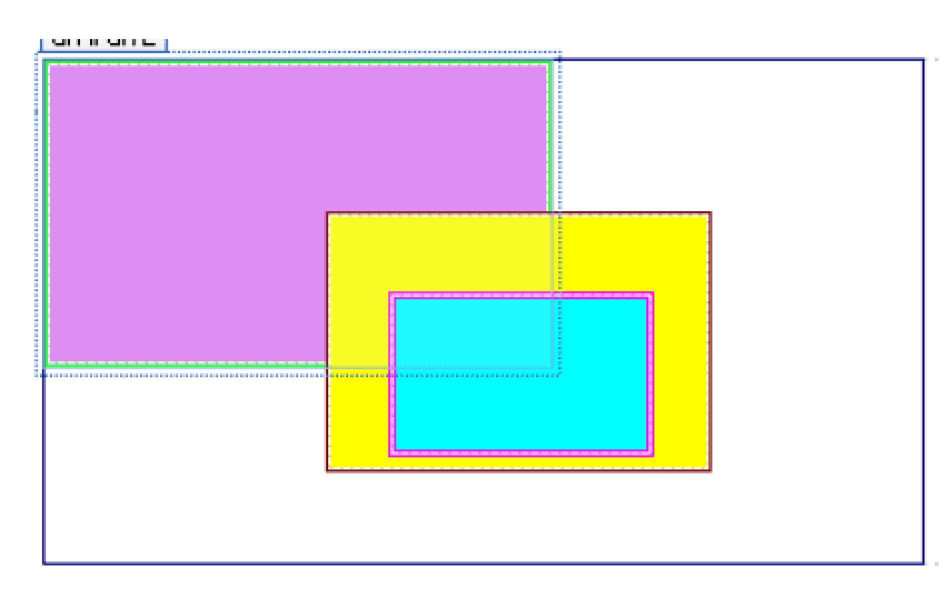


图 4-13　绝对定位的页面效果

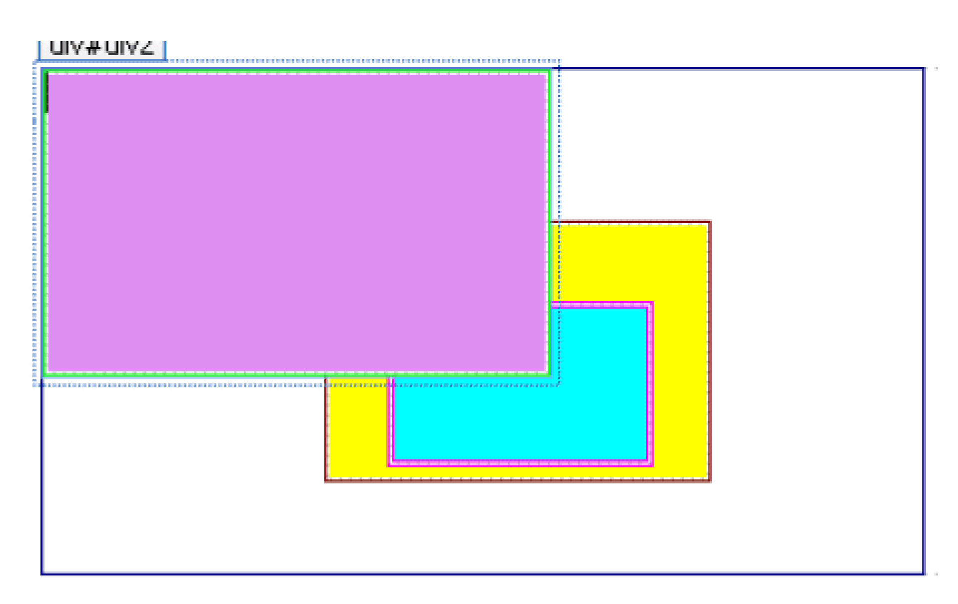


图 4-14　设置 z-index 属性后的效果

采用坐标定位的方式可以精确地将元素放在页面中相应的位置上，但是由于不同浏览器在显示方面存在的差异，也会给整体页面布局带来混乱的效果，解决这一问题的方法就是使用表格布局。

4.3.3　表格布局

利用表格可以将网页中的内容合理地放置在相应的区域，每个区域之间互不干扰。例如，设计一个表格用来布局网页首页，实现如图 4-15 所示的效果。

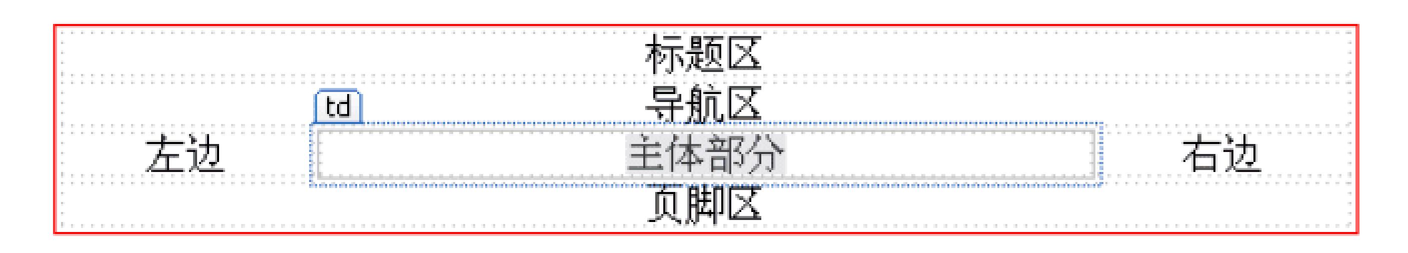


图 4-15　表格布局

例 4-5：创建表格，通过该表格实现如图 4-15 所示页面布局效果。

从图中可以看出，表格中定义了 1 个标题区，1 个导航区，1 个页脚区，中间又分成 3 个区，这就需要先创建一个 4 行 3 列的表格，然后再通过详细设置达到图中的效果。

(1) 启动 VWD 2010，打开网站 Chapter4，通过“添加新项”对话框在该网站中添加一个名为 Table.aspx 的 Web 窗体页。

(2) 切换到“设计”视图，将鼠标光标停在 div 标记内。选择“表”|“插入表”命令，打开“插入表格”对话框，定义表格大小为 4 行 3 列，指定宽度为 100%，边框值为 1，边框颜色为红色。如图 4-16 所示。

(3) 选中第一行的 3 个单元格，选择“表”|“修改”|“合并单元格”命令，将其合并。

(4) 用同样的方法合并第 2 行和第 4 行的 3 个单元格。

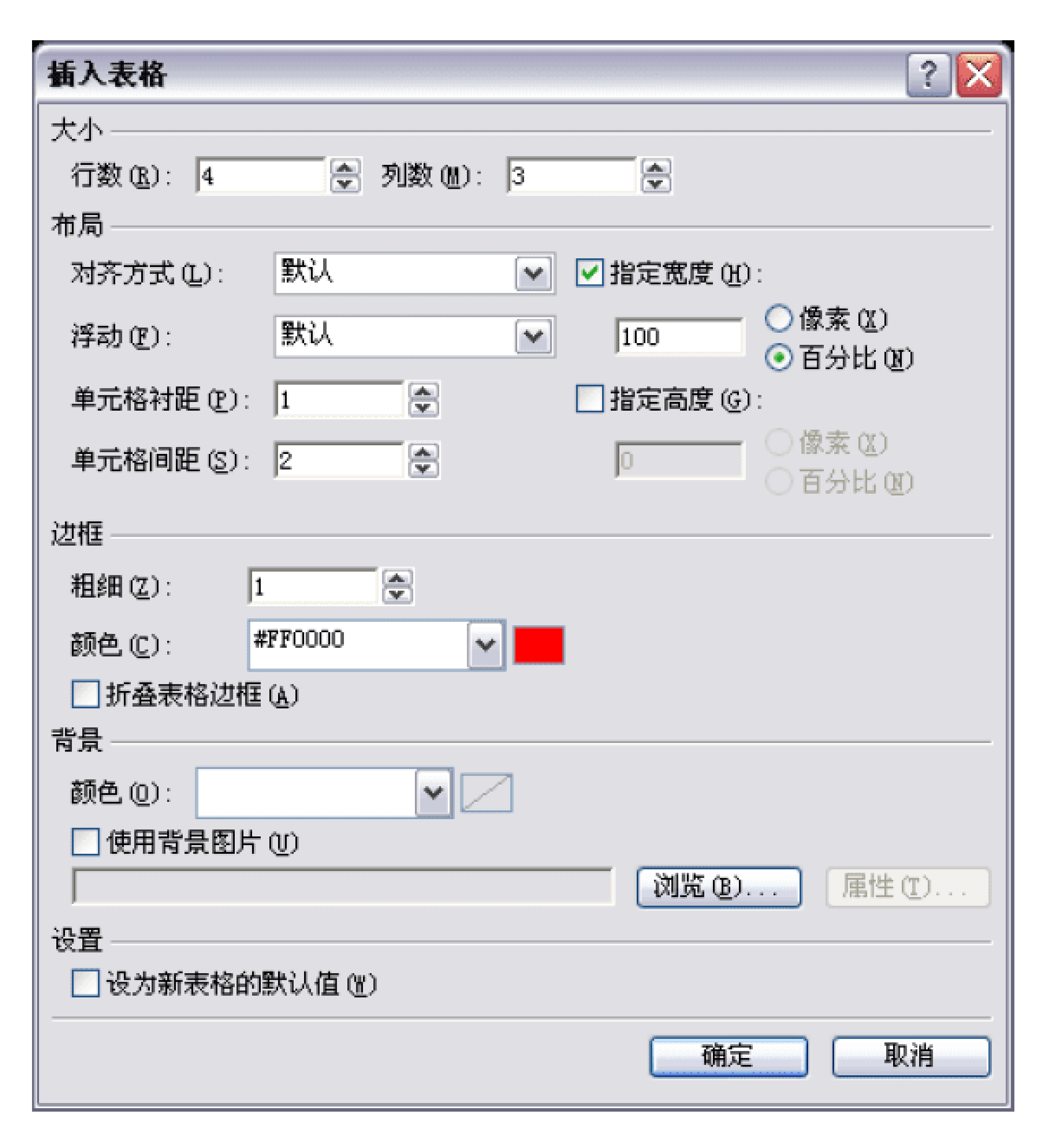


图 4-16　“插入表格”对话框

(5) 单击表格之外的其他空白处，此时“设计”视图的左上角将显示 body 标记，单击此处，将选择所有 body 区域中的元素，然后选择“格式”|“两端对其”|“居中”命令，使得页面中所有文字都居中显示，切换到“源”视图，可以看到 body 元素添加了如下样式。

```
<body style="text-align: center">
```

(6) 分别选中第 3 行的第 1 个单元格和第 3 个单元格，然后在“属性”面板中设置其 Width 属性为 20%。

(7) 分别在不同区域输入如图 4-15 所示的区域文本。

如果对以上布局不满意，还可以直接在“源”视图中修改响应的属性信息。表 4-2 中列出了表格中部分常用的属性。

表 4-2　常用的表格属性

属　性	说　　明
Border	表示边框宽度，如果设置为 0，表示无边框，此时默认 frame=void，rules=none；可以设置为大于 0 的值来显示边框，此时默认 frame=border，rules=all
Cellspacing	表示单元格间距(表格和 tr 之间的间隔)
Cellpadding	表示单元格衬距(td 和单元格内容之间的间隔)
Frames	表示如何显示表格边框，void：无边框(默认)；above：仅有顶部边框；below：仅有底部边框；hsides：仅有顶部和底部边框；vsides：仅有左右边框；lhs：仅有左边框；rhs：仅有右边框；box 和 border：包含全部 4 个边框
Rules	表示如何显示表格内的分隔线，all：显示所有分隔线；cols：仅显示列线；rows：仅显示行线；groups：仅显示组与组之间的分隔线

表格布局的最大优点是简单直观。但是如果将整个网页的元素都包含在表格内，则浏览器会将整个表格全部下载完毕后才显示表格中的内容，因此网页显示速度慢。此外，表格布局也不利于网页结构和表现的分离。解决该问题的方法就是网页整体采用 DIV 和 CSS 进行层布局，局部用表格进行布局。这是当前 Web 标准推荐的最佳布局方法。

4.3.4　DIV 和 CSS 布局

在传统的表格布局中，完全依赖于表格对象 TABLE，通常，在页面中绘制多个单元格，在表格中放置内容，通过表格的间距或者用无色透明的 GIF 图片来控制布局版块的间距，达到排版的目的。而以 DIV 对象为核心的页面布局中，通过层来定位，通过 CSS 定义外观，最大程度地实现结构和外观彻底分离的布局效果，因此习惯上对层布局又称为 DIV 和 CSS 布局。

层布局最核心的标签就是 DIV。DIV 是一个容器，在使用时以<div></div>形式存在。在 XHTML 中，每一个标签都可以称为容器，能够放置内容。但 DIV 是 XHTML 中专门用于布局设计的容器对象。

添加层的方法非常简单，直接在“源”视图中创建一对<div></div>标记即可。也可以通过“工具箱”中的 HTML 服务器控件 Div 来创建。

1. 盒子模型

自从 1996 年 CSS1 的推出，W3C 组织就建议把所有网页上的对象都放在一个盒子(box)中，设计师可以通过创建定义来控制这个盒子的属性，这些对象包括段落、列表、标题、图片以及层。盒子模型主要定义了 4 个区域，内容(content)、边框距(padding)、边界(border)和边距(margin)，如图 4-17 所示。

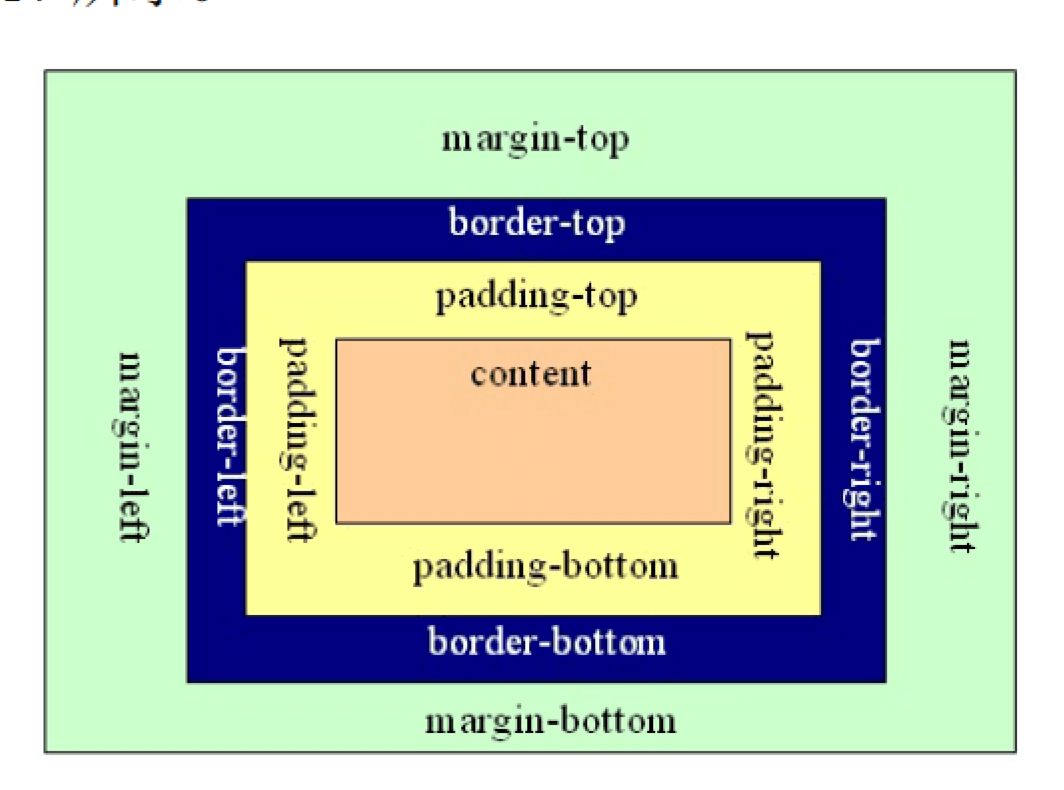


图 4-17　盒子模型

理解盒子模型就可以理解层与层之间定位的关系以及层内部的表达样式。其中 margin 属性负责层与层之间的距离，padding 属性负责内容和边框之间的距离。

下面的代码定义了盒子模型中的一些样式。

```
<style>
#sample
{
background-color: #FFCC00;
```

```
border-style: solid;
padding-bottom: 25px;
margin-bottom: 40px;
width: 80%;
}
</style>
```

2. 层的定位

在一个页面中定义多个层，会发现这些层自动排列在不同的行，而要真正实现左右排列，就要加入新的属性——float(浮动属性)。float 浮动属性是 DIV 和 CSS 布局中的一个非常重要的属性。大部分的 DIV 布局都是通过 float 的控制来实现的。具体参数如下。

- float:none 用于设置是否浮动。
- float:left 用于表示对象向左浮动。
- float:right 用于表示对象向右浮动。

例 4-6：利用层，创建一种左右上下分栏的样式。

(1) 启动 VWD 2010，打开网站 Chapter4，通过“添加新项”对话框在该网站中添加一个名为 div.aspx 的 Web 窗体页。

(2) 修改页面<head>和<body>内的代码如下所示：

```
<head runat="server">
    <style>
#left,#right{background-color:#eeeeee;border:1px solid #33ccff;height:200px; }
#left{width:180px; float:left; }
#bottom{ background-color:#eeeeee; border:1px solid #33ccff; height:50px; clear:both; }
    </style>
</head>
<body>
   <form id="form1" runat="server">
      <div id="left">当前层的 ID 是 left</div>
      <div id="right">当前层的 ID 是 right</div>
      <div id="bottom">当前层的 ID 是 bottom</div>
   </form>
</body>
```

(3) 切换到“设计”视图，其效果如图 4-18 所示。



图 4-18　左右上下分栏

3. 利用 DIV 和 CSS 实现页面布局

DIV 只是一个区域标识，划定了一个区域，要实现样式还需要借助于 CSS，这样的分离，使得 DIV 的最终效果是由 CSS 来编写的。CSS 可以实现左右分栏，也可以实现上下分栏，而表格则没有这么大的灵活性。CSS 与 DIV 的无关性，决定了 DIV 在设计上有极大的伸缩性，而不会被单元格固定的模式束缚。因此，实现网页布局，通常是先在网页中将内容用 DIV 标记出来，然后再用 CSS 来编写样式。

采用 DIV 和 CSS 布局之前，首先要分析网页有哪些内容块，以及每个内容块的含义，这就是所谓的网页结构。通常情况下，页面结构包含以下几块。

(1) 标题区(header)：用来显示网站的标志和站点名称等。

(2) 导航区(navigation)：用来表示网页的结构关系，如站点导航，通常放置主菜单。

(3) 主功能区(content)：用来显示网站的主题内容，如商品展示和公司介绍等。

(4) 页脚(footer)：用来显示网站的版权和有关法律声明等。

通常采用 DIV 元素来将这些结构先定义出来，类似这样：

```
<div id="header"></div>
<div id="globalnav"></div>
<div id="content"></div>
<div id="footer"></div>
```

然后在 CSS 样式表中定义每个元素 ID 的具体样式，从而控制整个页面的布局。例如，主功能区 ID 选择符的定义如下：

```
#content
{
    width: 740px;
    margin-top: 0px;
    margin-left:auto;
    margin-right:auto;
}
```

4.4　主题

学习了利用 CSS 控制页面上各元素的样式以及部分服务器控件的样式，但是有些服务器控件的属性无法通过 CSS 进行控制。为了解决这个问题，从 ASP.NET 2.0 开始就提供了一种称为“主题”的新方式，它可以保持网站外观的一致性和独立性，同时使页面的样式控制更加灵活方便，如动态实现不同用户界面的切换等。

4.4.1　主题概述

主题是定义页面和控件外观的文件的集合。它通常包含外观文件(扩展名为.skin)、级

联样式表文件(扩展名为.css)、图片和其他资源等的组合，但一个主题至少包含一个外观文件。

主题在 Web 站点的根文件夹中的特殊文件夹 App_Themes 中定义。在这个文件夹中需要创建定义实际主题的一个或多个子文件夹。在每个子文件夹中，可以有若干组成主题的文件。如图 4-19 所示的主题目录结构中创建了 3 个主题，分别是“主题 1”、“主题 2”和“主题 3”，“主题 1”中包含 1 个外观文件，“主题 2”中包含 2 个外观文件，“主题 3”中包含 1 个外观文件和 1 个样式表文件。

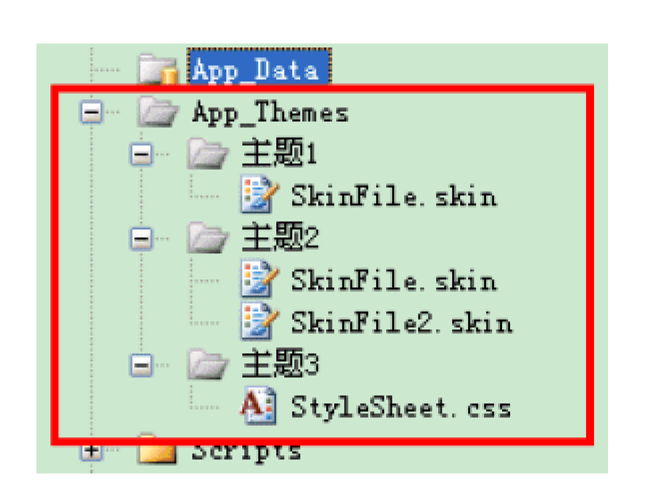


图 4-19　主题目录结构

1. 主题的类型

主题分为两大类型，一类是应用程序主题，另一类是全局主题。

- 应用程序主题是指保存在 Web 应用程序的 App_Themes 文件夹下的一个或多个主题文件夹，主题的名称就是文件夹的名称。
- 全局主题是指保存在服务器上，根据不同的服务器配置决定的，能够对服务器上所有 Web 应用程序起作用的主题文件夹。

一般情况下，很少用到全局主题，而本书所讲的主题也仅指应用程序主题，即保存在应用程序中 App_Themes 文件夹下的主题文件夹，简称主题。

ASP.NET 页面有两个不同的设置主题的属性，Theme(页主题)属性和 StyleSheetTheme(页的样式表主题)属性。这两个属性都使用在 App_Themes 文件夹中定义的主题。虽然一开始它们看起来非常相似，但是，在运行时它们的行为就不同了。StyleSheetThemes 在页面的生命周期中应用得非常早，在创建页面实例后不久就应用了。这意味着单个页面能通过在控件上应用内联属性来重写主题的设置。例如，带有将按钮的 BackColor 设置为紫色的外观文件的主题可以被页面中下面的控件声明重写：

```
<asp:Button ID="Button1" runat="server" Text="Button" BackColor="Blue" />
```

而 Theme 属性在页面的生命周期中应用的时间较晚，能有效地重写为单个控件自定义的任何属性。

由于 StyleSheetTheme 的属性能被页面重写，而 Theme 又能再次重写这些属性，两者用于不同的目的。如果想为控件提供默认设置则应该设置 StyleSheetTheme 属性，即 StyleSheetTheme 能为控件提供默认值，然后又可以在页面级重写。如果想强制应用控件的外观则应使用 Theme 属性，因为 Theme 中的设置不能再重写，而且它有效地重写了任何自定义设置，因此能确保控件的外观就是在主题中定义的样子。

如果由于某种原因不想向特定控件应用外观，可以通过设置控件的属性

EnableTheming 来禁用外观，如下所示：

```
<asp:Button ID="Button1" runat="server" EnableTheming="False" Text="Button" />
```

由于 EnableTheming 被设置为 False，因此就不会向控件应用外观。而仍然会应用主题的 CSS 文件中的 CSS 设置。

2. 外观文件

外观文件是主题的核心文件，也称为皮肤文件，专门用于定义服务器控件的外观。在主题中可以包含一个或多个外观文件，外观文件的后缀名为.skin。

在控件外观设置中，只能包含主题的属性定义，如样式属性、模板属性和数据绑定表达式等，不能包含控件的 ID，如 Label 控件的外观设置代码如下：

```
<asp:Label runat="server" BackColor="Blue" Font-Names="Arial Narrow" />
```

这样，一旦将该外观应用到 Web 页面中，所有的 Label 控件都将显示外观所设置的样式。

右击某一个“主题”文件夹，从弹出的快捷菜单中选择“添加新项”命令，在弹出的“添加新项”对话框中选择“外观文件”选项，即可添加一个外观文件。

3. 级联样式表文件

主题中可以包含一个或多个 CSS 文件，一旦 CSS 文件被放在主题中，则应用时无须再在页面中指定 CSS 文件链接，而是通过设置页面或网站所使用的主题即可，当主题得到应用时，主题中的 CSS 文件会自动应用到页面中。

右击某一个“主题”文件夹，从弹出的快捷菜单中选择“添加新项”命令，在弹出的“添加新项”对话框中选择“样式表文件”选项，即可添加样式表文件。

4.4.2　创建和应用主题

要创建一个主题，需要做下列事情。

- 如果站点中还没有 App_Themes 文件夹，则首先要创建该文件夹。
- 对于要创建的每个主题，用主题的名称创建一个子文件夹。
- 可选地，创建一个或多个将成为主题一部分的 CSS 文件。虽然根据主题命名 CSS 文件有助于标识正确的文件，但并不要求一定要这样做。添加到主题的文件夹中的任何 CSS 文件都会在运行时自动添加到页面中。
- 可选地，向主题文件夹中添加一个或多个图像。CSS 文件应当用稍后将介绍的相对路径来引用这些图像。
- 可选地，向主题文件夹中添加一个或多个外观文件。外观允许为之后要在运行时应用的特定控件定义单个属性(如 ForeColor 和 BackColor)。

执行了这些步骤以后，就能将站点或单个 Web 页面配置为使用此主题。

技巧：

如果站点中没有 App_Themes 文件夹，可以在“解决方案资源管理器”中，右击项目名，从弹出的快捷菜单中选择“添加” | “添加 ASP.NET 文件夹” | “主题”命令，创建 App_Themes 文件夹，并在该文件夹下生成一个默认名为“主题 1”的文件夹。

1. 在主题中定义外观

skin 文件必须直接在主题的文件夹中创建。不能像存储主题的图像那样把它们存储在一个子文件夹中。

在外观文件中，系统没有提供控件属性设置的智能提示功能，这使得定义自己的控件及其属性比较困难。如果要使用 VWD 的智能提示功能，可做如下设置。

(1) 选择“工具” | “选项”命令，打开“选项”对话框。

(2) 展开“文本编辑器”选项，然后选择“文件扩展名”。

(3) 在右侧的“扩展名”文本框中输入 skin，然后从“编辑器”下拉列表中选择“用户控件编辑器”选项。

(4) 单击“添加”按钮，然后单击“确定”按钮完成设置，如图 4-20 所示。

设置完以后，当再次打开一个 skin 文件时，就会出现智能提示功能了。

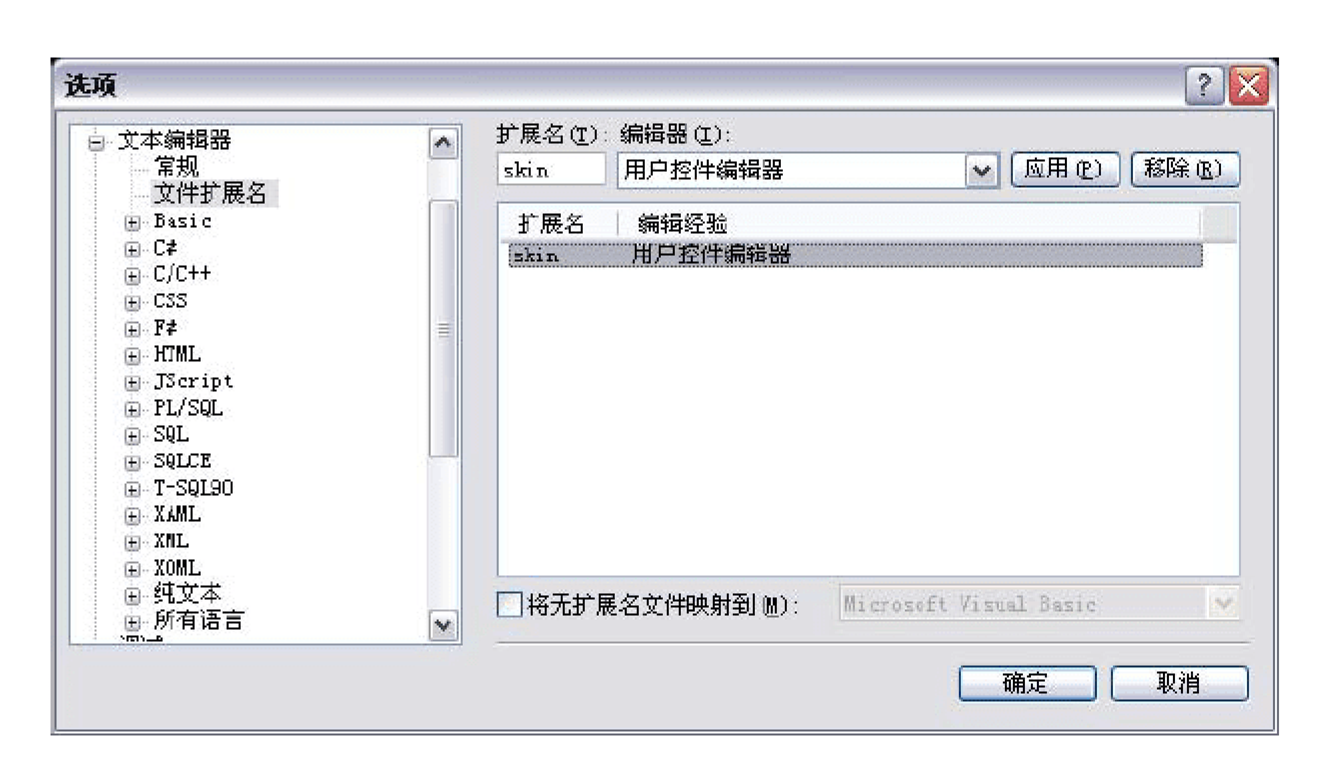


图 4-20　“选项”对话框

例 4-7：创建一个包含一些简单外观的主题，这些外观用于定义控件的外观。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 在“解决方案资源管理器”中，右击项目名程，从弹出的快捷菜单中选择“添加” | “添加 ASP.NET 文件夹” | “主题”命令，系统将创建名为 App_Themes 的文件夹和名为“主题 1”的子文件夹。

(3) 右击“主题 1”文件夹，从弹出的快捷菜单中选择“添加新项”命令，添加一个新的外观文件 SkinFile.skin。

(4) 新添加的文件中包括一段外观文件编写的说明文字和两个示例，如下所示：

```
<%--
默认的外观模板。以下外观仅作为示例提供。
1. 命名的控件外观。SkinId 的定义应唯一，因为在同一主题中不允许一个控件类型有重复的
SkinId。
```

```
<asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White" >
    <AlternatingRowStyle BackColor="Blue" />
</asp:GridView>
2. 默认外观。未定义 SkinId。在同一主题中每个控件类型只允许有一个默认的控件外观。
<asp:Image runat="server" ImageUrl="~/images/image1.jpg" />
--%>
```

(5) 按照以上示例，添加如下代码，定义 Label 和 Button 控件的外观。

```
<asp:Label runat="server" ForeColor="red" Font-Size="14pt" Font-Bold="true" />
<asp:Button runat="server" Borderstyle="Solid" Borderwidth="2px" Bordercolor="Blue"
Backcolor="yellow"/>
```

(6) 保存该外观文件。通过“添加新项”对话框添加一个名为 ThemeTest.aspx 的网页，切换到“设计”视图，添加 1 个 Label 控件和 1 个 Button 控件。

(7) 在“属性”面板中选择 Document 元素，设置 Theme 属性为“主题 1”，切换到“源”视图中，会发现代码第 1 行的@ Page 指令中添加了下面的属性：

```
<%@ Page … Theme="主题 1"%>
```

(8) 编译并运行程序，在默认浏览器中打开该页面，查看控件的效果，如图 4-21 所示。

2. 使用 SkinID 属性

如果希望某些控件的外观和页面中具有相同类型的其他控件的外观不一样，则可以在.skin 文件中给特定的控件添加一个 SkinID 属性，来看下面的例子。

例 4-8：在例 4-7 的基础上增加一个按钮控件的外观定义并指定 SkinID 属性，然后在测试页面中应用该特定外观。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 打开外观文件 SkinFile.skin。

(3) 添加一个按钮的外观定义，并为其设置 SkinID 属性，如下所示：

```
<asp:Button runat="server" SkinID="GreenBtn"  Font-Size="14pt" Borderstyle="dotted"
Borderwidth="2px" Bordercolor="red" Backcolor="Green"/>
```

(4) 在 ThemeTest.aspx 页面中在添加另一个 Button 控件，设置新添加的 Button 控件的 SkinID 属性为 GreenBtn。显示效果如图 4-22 所示。

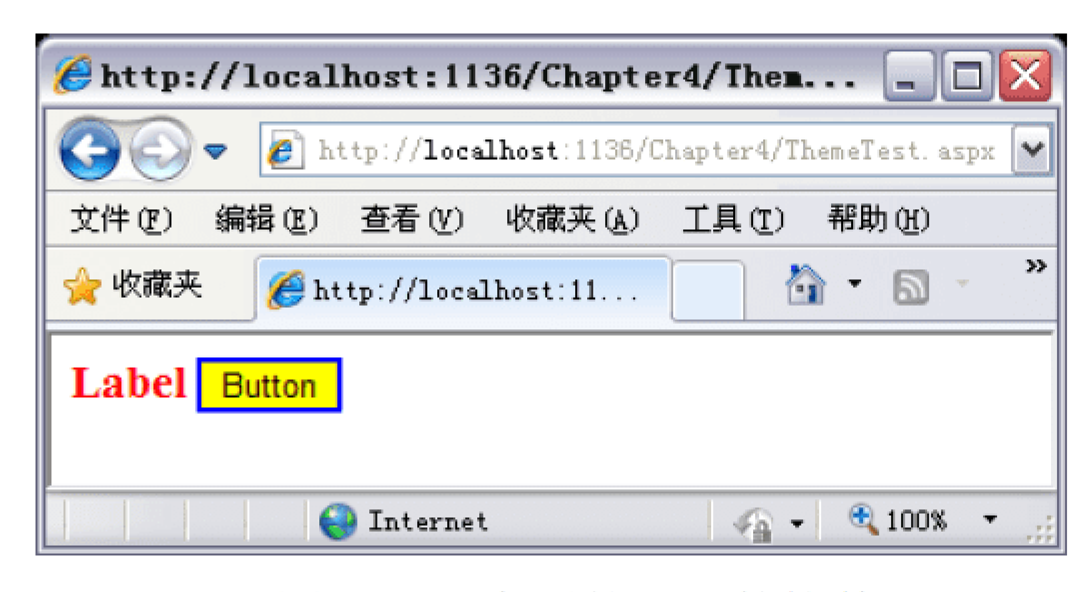


图 4-21　应用外观后的控件



图 4-22　应用 SkinID 属性的控件

3. 在主题中定义样式表

除了外观文件，在主题中还可以定义.css 文件。然后在网页文件中设置 StyleSheetTheme 属性为定义的主题即可。

例 4-9：在网页文件中同时使用外观文件和样式表文件。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 右击“主题 1”文件夹，从弹出的快捷菜单中选择“添加新项”命令，添加一个样式表文件 StyleSheet1.css。

(3) 在 StyleSheet1.css 样式文件中添加如下代码，定义 h2 的样式：

```
h2
{
    border-style: dashed;
    font-size: 1.5em;
    padding-bottom: 0px;
    margin-bottom: 2px;
    color: #FF00CC;
}
```

(4) 在 ThemeTest.aspx 页面中添加如下<h2>标记的元素。

```
<h2>欢迎光临金百合拉丁舞培训学校网站</h2>
```

(5) 修改当前页面的 Document 中属性 StyleSheetTheme 的值为“主题 1”。

(6) 编译并运行程序，在浏览器中即可看到引入外观和样式表文件后的最终显示效果，如图 4-23 所示。

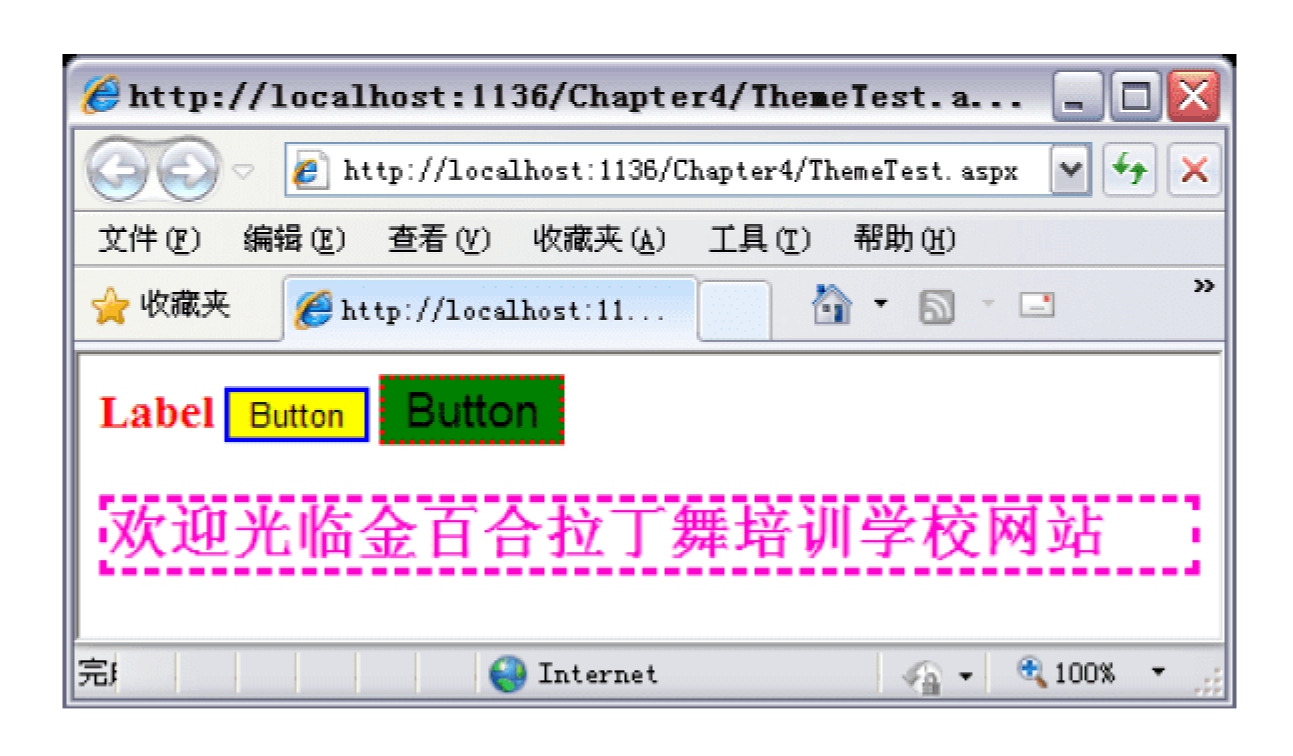


图 4-23　引入外观和样式后的页面效果

4.4.3　主题的应用级别

创建了主题之后，可以定制如何在应用程序中使用主题，方法是将主题作为自定义主题与网页文件关联，或者将主题作为样式表主题与网页文件关联。样式表主题和自定义主题都使用相同的主题文件，但是样式表主题在网页文件的控件和属性中的优先级最低。在

ASP.NET 中，主题的优先级顺序如下。

(1) 主题设置，包括 Web.config 文件中设置的主题。

(2) 本地网页文件的样式属性设置。

(3) 样式表主题设置。

在这里，如果选择使用样式表主题，则在网页文件中本地声明的任何样式信息都将覆盖样式表主题的属性。同样，如果使用自定义主题，则主题的属性将覆盖本地网页文件中设置的任何样式内容，以及任何样式表主题中的任何内容。

有 3 个不同的选项可以向 Web 站点应用主题，分别是：在 Page 指令中的页面级、在站点级修改 web.config 文件、通过程序来设置主题。

1. 在页面级设置主题

前面的例 4-7 和例 4-8 中就是使用这种方式来应用主题的。在页面级设置 Theme 属性或 StyleSheetTheme 属性很容易，只需设置页面的 Page 指令中的相关属性即可。

```
<%@ Page Language="C#" AutoEventWireup="false" CodeFile="Default.aspx.cs"
        Inherits="_Default" Theme="主题 1" StyleSheetTheme="主题 1" %>
```

注意：

用 StyleSheetTheme 替换 Theme 来应用一个主题，该主题的设置可以由单个页面重写。

2. 在站点级设置主题

如果要在整个 Web 站点中强制应用同一个主题，可以在 web.config 文件中设置主题。要做到这一点，需要将一个 theme 属性添加到<system.web>元素内的<pages>元素中。

```
<pages theme="主题 1">
   ...
</pages>
```

提示：

确保全部用小写字母输入 theme，因为 web.config 文件中的 XML 是区分大小写的。

3. 通过程序来设置主题

设置主题的第三种也是最后一种方式是通过代码来编程设置的。由于主题的工作方式，需要在页面生命周期的早期完成这一工作。通常是在 PreInit 事件中，通过 Page 对象的 Theme 属性来设置主题。

4.4.4　扩展主题

除了 CSS 文件与外观以外，主题还可以包含图像。主题图像最普遍的用法是从 CSS 中引用它们。要充分利用图像，就要了解 CSS 如何引用图像。

例 4-10：向主题中添加图像。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 右击“主题 1”文件夹，从弹出的快捷菜单中选择“新建文件夹”命令，新建一个文件夹，修改其名称为 images，在 images 文件夹上右击，选择“添加现有项”命令，添加一个图片 bg.jpg 作为主题 1 的背景图片。

(3) 为了引用 bg.jpg 文件，可以向 StyleSheet1.css 中添加如下 CSS 样式。

```
body
{
  background-image: url(images/bg.jpg);
}
```

(4) 此时在浏览器中打开 ThemeTest.aspx 页面，可以看到含有背景图片的 Web 页面了，如图 4-24 所示。

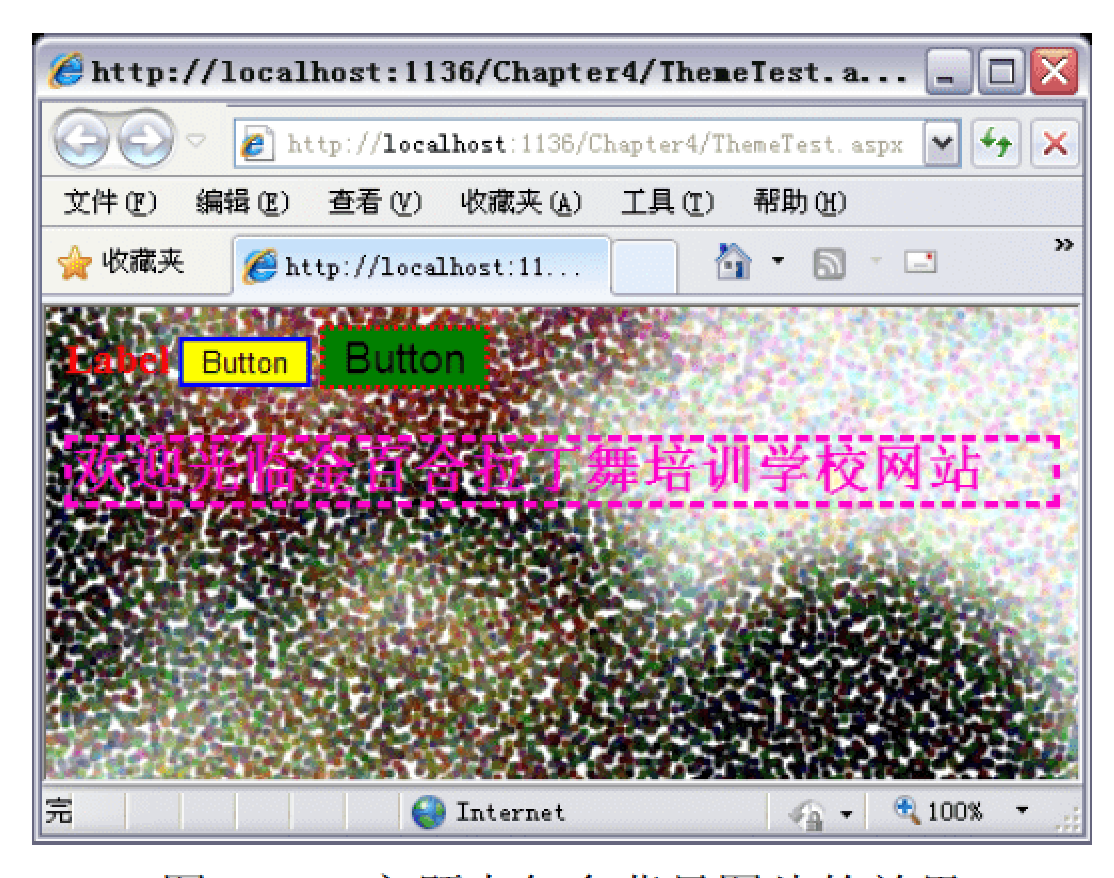


图 4-24　主题中包含背景图片的效果

说明：

在设计时，除非给出一个以指示站点根文件夹的正斜杠(/)开头的路径，否则 CSS 选择器引用的图像会相对于 CSS 文件的当前位置搜索。

4.4.5　动态切换主题

动态切换主题是指在运行时切换主题。例如，可以允许用户用喜欢的颜色和布局选择主题。由于使用的是在运行时向页面应用主题的方式，因此需要在页面的生命周期较早的时候设置主题，即在 PreInit 事件中设置。

为了允许用户修改主题，可以提供给他们一个下拉菜单，当用户修改列表中的活动选项时，该菜单自动向服务器发起回发请求。在服务器端，就会得到从列表中选择的主题，将它应用到页面上，然后将选项存储在 cookie 中，以便下次访问时检索它。

例 4-11：让用户选择自己喜欢的主题，实现动态换肤功能。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 右击“主题 1”文件夹，从弹出的快捷菜单中选择“复制”命令，然后再次右击，选择“粘贴”命令，得到“副本 主题 1”，通过“重命名”命令将其命名为“主题 2”。

(3) 用同样的方法，再复制一份主题，并将复制后的主题命名为“主题 3”。

(4) “主题 2”定义为“浪漫小屋”，所以背景图片和控件的外观都以粉色和橘黄色为主，修改主题 2 中的外观文件 SkinFile.skin，添加如下代码：

```
<asp:Label runat="server" BackColor="Fuchsia" BorderColor="White" Borderstyle="dotted"
            Font-Bold="True" Font-Size="Larger" ForeColor="Blue"></asp:Label>
<asp:DropDownList runat="server" BackColor="#FFFF99"
            Font-Bold="True" Font-Size="Large" ForeColor="#FF0066"></asp:DropDownList>
<asp:Calendar runat="server" BackColor="#FFFFCC"
            BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest"
            Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399" Height="200px"
            ShowGridLines="True" Width="220px">
            <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
            <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
            <OtherMonthDayStyle ForeColor="#CC9966" />
            <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
            <SelectorStyle BackColor="#FFCC66" />
            <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
                ForeColor="#FFFFCC" />
            <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
        </asp:Calendar>
```

说明：

为了让用户可以动态地切换主题，会在测试页面中添加 DropDownList 控件，所以上述外观文件中定义了 DropDownList 控件的外观样式。另外，还定义了 Calendar 控件的外观，这个主要用来显示不同的效果，以区分不同的主题。

(5) 主题 2 样式表文件中只定义背景图片，无须修改，只须替换一下图片即可。

(6) “主题 3”定义为“鸟语花香”，所以背景图片和控件的外观都以青春和绿色为主，在主题 3 的外观文件中添加如下代码：

```
    <asp:Label runat="server" BackColor="#FFFF66" ForeColor="#0066FF" Font-Bold="True"
Font-Size="Large" BorderStyle="Groove" />
    <asp:DropDownList runat="server"    Font-Bold="True"
            BackColor="#FFFFCC" ForeColor="Lime" Font-Size="Large"/>
    <asp:Calendar runat="server" BackColor="White"
            BorderColor="#66FF33" BorderWidth="1px" CellPadding="1"
            DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
            ForeColor="#003399" Height="200px" Width="220px">
            <DayHeaderStyle BackColor="#99FF33" ForeColor="#99FF66" Height="1px" />
            <NextPrevStyle Font-Size="8pt" ForeColor="#CCCCFF" />
            <OtherMonthDayStyle ForeColor="#999999" />
```

```
        <SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99" />
        <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />
        <TitleStyle BackColor="#009900" BorderColor="#3366CC" BorderWidth="1px"
            Font-Bold="True" Font-Size="10pt" ForeColor="#CCCCFF" Height="25px" />
        <TodayDayStyle BackColor="#99CCCC" ForeColor="White" />
        <WeekendDayStyle BackColor="#66FF66" />
    </asp:Calendar>
```

(7) 主题 3 的样式表文件中也只需替换一下背景图片即可。

(8) 通过“添加新项”对话框添加一个名为 ThemeTest2.aspx 的测试页面，切换到“设计”视图，添加一个 Label 控件、一个 DropDownList 控件和一个 Calendar 控件。

(9) 设置 Label 控件的 Text 属性为“请选择主题：”，为 DropDownList 控件的 Items 属性添加几个选择，设置 AutoPostBack 属性为 True，生成的代码如下：

```
    <asp:Label ID="Label1" runat="server" Text="请选择主题："></asp:Label>
    <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True">
        <asp:ListItem Value="主题 1">默认主题</asp:ListItem>
        <asp:ListItem Value="主题 2">浪漫小屋</asp:ListItem>
        <asp:ListItem Value="主题 3">鸟语花香</asp:ListItem>
    </asp:DropDownList>
```

(10) 为 DropDownList 控件添加 SelectedIndexChanged 事件处理程序，代码如下：

```
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        HttpCookie myTheme = new HttpCookie("userTheme");
        myTheme.Expires = DateTime.Now.AddMonths(3);
        myTheme.Value = DropDownList1.SelectedValue;
        Response.Cookies.Add(myTheme);
        Response.Redirect(Request.Url.ToString());
}
```

(11) 当页面加载时将需要再次从列表中预先选择恰当的项，以显示正确的主题。进行此操作的最佳位置是在 Page 类的 Load 事件中。添加处理程序的代码如下：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            string selectedTheme = Page.Theme;
            HttpCookie myTheme = Request.Cookies.Get("userTheme");
            if (myTheme != null)
            {
                selectedTheme = myTheme.Value;
            }
```

```
            if (!string.IsNullOrEmpty(selectedTheme) &&
                    DropDownList1.Items.FindByValue(selectedTheme) != null)
            {
                DropDownList1.Items.FindByValue(selectedTheme).Selected = true;
            }
        }
    }
```

(12) 正如前面所提到的，主题需要在 PreInit 事件(该事件在页面生命周期的早期发生)中设置。在该事件内可以查看带选中主题的 cookie 是否存在。如果存在，就可以用它的值设置恰当的主题，代码如下：

```
    protected void Page_PreInit(object sender, EventArgs e)
    {
        HttpCookie preferredTheme = Request.Cookies.Get("userTheme");
        if (preferredTheme != null)
        {
            Page.Theme = preferredTheme.Value;
        }
        else
        {
            Page.Theme = "主题 2";//默认使用主题 2
        }
    }
```

(13) 编译并运行程序，在浏览器中打开 ThemeTest2.aspx 页面，通过下拉列表选择不同的主题，再效果如图 4-25 所示。

技巧：

在外观文件中定义控件的外观时，可以在普通页面中添加一个要设置外观的控件，在“设计”视图中通过“属性”面板对其进行格式化设置，然后将生成的代码复制到.skin 文件中，再删除 ID 属性即可。



图 4-25　动态切换主题效果

4.5　母版页

在构建 Web 站点时，应该使布局和行为尽可能保持一致。而且有很多元素会出现在每一个页面中，如站点标题、公共导航以及版权信息等，这些元素的一致布局会让用户知道自己始终是在同一个站点中。虽然这些元素可以通过在 XHTML 中使用包含文件来构建，但 ASP.NET 4 和 VWD 2010 提供了更加健壮的母版页技术来实现。

母版页的最大好处是可以在单个地方定义站点中所有页面的全局外观。这样，当需要修改站点的布局(如要把菜单从左边移到右边)时，只需修改母版页，基于此类母版页的页面就会自动进行相应的修改。

4.5.1　母版页概述

母版页是用于设置页面外观的模板，是一种特殊的 ASP.NET 网页文件，同样也具有其他 ASP.NET 文件的功能，如添加控件、设置样式等，只不过它的扩展名是.master。在母版页中，界面被分为公用区和可编辑区，公用区的设计方法与一般页面的设计方式相同，可编辑区则用 ContentPlaceHolder 控件预留出来。

引用母版页的.aspx 页面称为内容页，在内容页中，母版页的 ContentPlaceHolder 控件预留的可编辑区会被自动替换为 Content 控件，开发人员只需在 Content 控件区域中填充内容即可，在母版页中定义的其他标记将自动出现在引用该母版页的内容页中，母版页的部分以灰色显示，表示不能编辑这些内容。

基于一个母版页可以创建一个或多个内容页，使用母版页可以统一管理和定义具有相同布局风格的页面，给网页设计和修改带来极大的方便。使用母版页具有如下优点。

- 使用母版页可以集中处理页的通用功能，以便可以只在一个位置进行更新。
- 使用母版页可以方便地创建一组控件和代码，并将结果应用于一组新的页面。
- 通过允许控制占位符控件的呈现方式，母版页可以在细节上控制最终页的布局。
- 母版页提供一个对象模型，使用该对象模型可以从各个内容页自定义母版页。

在使用母版页时，母版页中使用的图片和超链接应尽量使用服务器端控件来实现，如 Image 和 HyperLink 控件。即使控件不需要服务器代码也是如此，因为将设计好的母版页或内容页移动到另一个文件夹时，如果使用的是服务器控件，即使不改变服务器控件的 URL，ASP.NET 也可以正确解析，并自动将其 URL 改为正确的位置，但是如果使用了普通 HTML 标记，那么 ASP.NET 将无法正确解析这些标记的 URL，从而导致图片不能显示或链接失败，给维护带来极大麻烦。

4.5.2　创建母版页

当创建新的 Web 站点时，总是先添加作为所有其他页面基础的母版页。即使站点中只有少数几个页面，母版页仍然可以帮助确保整个站点拥有一致的外观。

在某种程度上，母版页看起来就像正常的 ASPX 页面。创建母版页的方法也和创建一般页面的方法非常相似，区别是母版页无法单独在浏览器中查看，必须通过创建内容页才能浏览。

1. 创建母版页

下面这个例子是一个很常见的布局，母版页中包含一个标题、一个导航菜单和一个页脚，这些内容将在站点的每个页面中出现。在母版页中包含两个个内容占位符，其中导航菜单有默认内容，主区域为空，这是母版页中的一个可变区域，可以使用内容页中的信息来替换此区域。

例 4-12：创建母版页。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 在“解决方案资源管理器”中右击网站的名称，从弹出的快捷菜单中选择“添加新项”命令，在打开的“添加新项”对话框中选择“母版页”模板，添加名为 MasterPage.master 的母版页。

(3) 观察母版页的源代码，在页面的顶部是一个@ Master 声明，而不是通常在 ASP.NET 页面中看到的@ Page 指令，它也有 CodeFile 和 Inherits 属性，如下所示：

```
    <%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>
```

(4) 此外，页面的主体部分还包含一个 ContentPlaceHolder 控件，这是母版页中的一个区域，其中的可替换内容将在运行时由内容页合并。

说明：

为了方便母版页的编辑，通常情况下先将 ContentPlaceHolder 控件删除，母版页编辑完成后再放置 ContentPlaceHolder 控件，下面的步骤将采用这种方法布局。

(5) 在母版页的<form>标记之间添加下面的代码，替换掉<div>标记与创建母版页时 VWD 添加的 ContentPlaceHolder。

```
        <form id="form1" runat="server">
          <div id="PageWrapper">
            <div id="top" align="center"><h1>欢迎光临金百合拉丁舞培训学校</h1></div>
            <div id="menu" align="right">
                <asp:ContentPlaceHolder id="menuContent" runat="server">
                    <a href="Index.aspx">首页</a> <a href="Info.aspx">学校简介</a> <a
href="About.aspx">关于我们</a>
                </asp:ContentPlaceHolder>
            </div>
            <div id="main">
              <asp:ContentPlaceHolder ID="mainContent" runat="server">
              </asp:ContentPlaceHolder>
            </div>
```

```
            <div id="footer" align="center" style="color:Gray">版权所有(C)金百合拉丁舞培训学校
2012.07.03</div>
          </div>
      </form>
```

(6) 接下来，将母版页切换到“设计”视图中，然后从“解决方案资源管理器”中将样式表文件 StyleSheet.css 拖到母版页上。此时，<head>标记内将添加一个指向 CSS 文件的<link>标记：

```
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
```

(7) 至此，完成母版页的创建，该母版页的设计视图效果如图 4-26 所示。

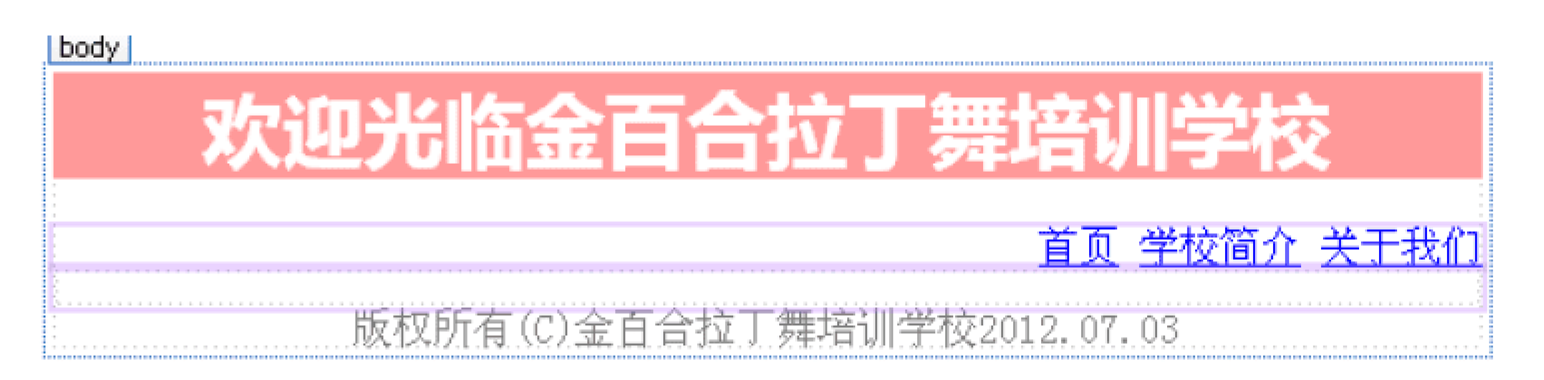


图 4-26　母版页设计效果

在下一小节，将介绍如何将基于该母版页创建内容页。

2. 母版页详解

已经创建了带有主内容占位符的母版页。切换到母版页的“源”视图，将发现页面的页头 head 部分也有一个 Content PlaceHolder。

```
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
```

每创建一个新的母版页时都会自动添加此占位符，在内容页中可以用它来添加页面特有的位于页面的<head>标记之间的内容，如 CSS(包括内嵌样式表和外部样式表)和 JavaScript。

母版页中名为 menuContent 的 ContentPlaceHolder 包含 3 个超链接，这是可以作为内容页的默认新项，当基于该母版页新建页面时，内容页既可以重写这部分内容，也可以不重写。

3. 嵌套母版页

母版页也可以嵌套。嵌套母版页是基于另一个母版页的母版页。内容页面则可以基于嵌套母版页。如果有一个目标为不同区域仍然需要共享相同外观的 Web 站点，采用嵌套母版页就比较有用。例如，有一个公司网站，分为各个部门。外部母版页定义站点的全局外

观，包括公司 logo 和其他品牌元素等。然后不同的部门又可以创建自己的嵌套母版页，这样各部门就能向它们在站点的部分中加上自己的身份标识。

嵌套母版页的创建很简单。当添加母版页时选中“选择母版页”复选框即可，就像后面介绍的添加内容页一样。然后，在内容页中要重写的位置将<asp:ContentPlaceHolder>控件添加到<asp:Content>控件中。

4.5.3　创建内容页

母版页如果没有内容页来使用它，那就没有任何用处。通常，仅有少量几个母版页，却可以有很多内容页。为了将一个内容页基于一个母版页，在添加新网页到站点时，就指定母版页，因此，只需选中“添加新项”对话框底部的“选择母版页”复选框即可。当然，也可以在直接在页面的@Page 指令中设置 MasterPageFile 属性。

内容页中只能含有映射到母版页中的<asp:ContentPlaceHolder>控件的<asp:Content>控件。而这些控件又可以包含标准标记，如 HTML 元素和服务器控件声明。

提示：

因为内容页中的整个标记需要用<asp:Content>标记括起来，所以不太容易将现有 ASPX 页面转换为内容页。通常是将要保留的内容复制到剪贴板上，删除原页面，然后基于母版页添加新页面，添加了该页面后，再把剪贴板上的内容粘贴到<asp:Content>标记内。

例 4-13：基于例 4-12 创建的母版页，创建相应的内容页。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 通过“添加新项”对话框添加 3 个新页面 Index.aspx、Info.aspx、About.aspx。添加上述 3 个页面时，需要选中“添加新项”对话框中的“选择母版页”复选框，并在弹出的“选择母版页”对话框中选择例 4-12 中创建的母版页 MasterPage.master。

(3) 基于母版页新建的网页初始代码如下所示：

```
    <%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Index.aspx.cs" Inherits="Index" %>
    <asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    </asp:Content>
    <asp:Content ID="Content2" ContentPlaceHolderID="menuContent" Runat="Server">
    </asp:Content>
    <asp:Content ID="Content3" ContentPlaceHolderID="mainContent" Runat="Server">
    </asp:Content>
```

(4) 指向 ContentPlaceHolder 的 Content 控件的 ContentPlaceHolderID 属性是在母版页中定义的。ContentPlaceHolderID 为 head 的占位符就是用来添加页面特有的位于<head>标记之间的内容的，本例中我们对此占位符不做任何修改。只设置菜单内容和主内容区域。

(5) 切换到 Index.aspx 页面的“设计”视图，单击 menuContent 控件右侧的小三角按钮，打开“Content 任务”面板，并选择“默认为母版页的内容”选项，此时将弹出“确认”对

话框，提示用户如果使用母版页的内容将从网页中删除此区域中的所有内容，单击“是”按钮，如图 4-27 所示。

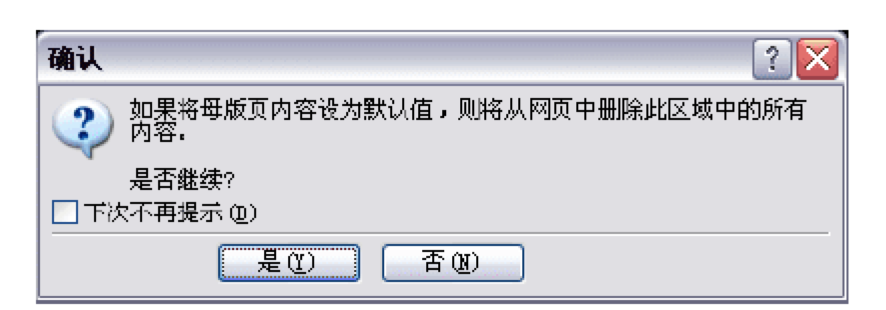


图 4-27　“确认”对话框

说明：

将默认值设置为母版页的内容之后，还可以通过“Content 任务”面板中的“创建自定义内容”选项来创建自己的内容

(6) 分别在 3 个页面的 mainContent 区域添加不同的内容以区分不同的页面。

(7) 编译并运行程序，当在浏览器中请求基于母版页的页面时，服务器会阅读内容页与母版页，将两者合并，然后将最终结果发送给浏览器，效果如图 4-28 所示。



图 4-28　页面运行效果

4.5.4　从内容页访问母版页中的成员

在内容页中可以通过编程的方式访问母版页中的成员，包括母版页上的任何公共属性或方法以及任何控件。要实现内容页对母版页中定义的属性或方法进行访问，则该属性或方法必须声明为公共成员(public)，也可以对母版页动态地进行访问。

1. 访问母版页的公共成员

要想在内容页中访问母版页上的属性，必须在母版页上先创建一个属性，下面举例说明。

例 4-14：在母版页中定义一个属性，然后在基于该母版页创建的内容页中访问该属性。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 打开母版页 MasterPage.master 的后台代码文件 MasterPage.master.cs。

(3) 在类定义中创建名为 strName 的属性，并在视图状态中存储该属性的值。添加的代码如下：

```
    public string strName
    {
        get{return (string)ViewState["myName"];}
        set { ViewState["myName"] = value; }
    }
```

(4) 添加页面的 Init 事件处理程序代码，如下：

```
    void Page_Init(Object sender, EventArgs e)
    {
        this.strName = "小石头";
    }
```

(5) 接下来，将在例 4-13 中创建的内容页 About.apsx 中访问该变量。切换到 About.aspx 页的“源”视图。在页面顶部的@ Page 指令下面添加如下@ MasterType 指令：

```
<%@ MasterType virtualpath="~/MasterPage.master" %>
```

此指令的作用是将内容页的 Master 属性绑定到 MasterPage.master 页。

(6) 切换到该页的“设计”视图，在 mainContent 区域中添加一个 Label 控件、一个 TextBox 控件和一个 Button 控件，设置 Button 控件的 Text 属性为“提交”。

(7) 在 About.aspx 页面的 Load 事件处理程序中添加如下代码：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "这是获取到母版页中的变量值：" + Master.strName;
    }
```

(8) 为按钮控件添加单击事件处理程序，代码如下：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (TextBox1.Text != "")
        {
            Master.strName = TextBox1.Text;
            Label1.Text = "这是获取到母版页中的变量新值：" + Master.strName;
        }
        else
        {
            string info = "alert(\"请输入新的变量值！\"); ";
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", info, true);
        }
    }
```

(9) 测试内容页，在浏览器中打开页面 About.aspx，初次加载页面显示母版页中变量的初始值“小石头”，如图 4-29 所示。

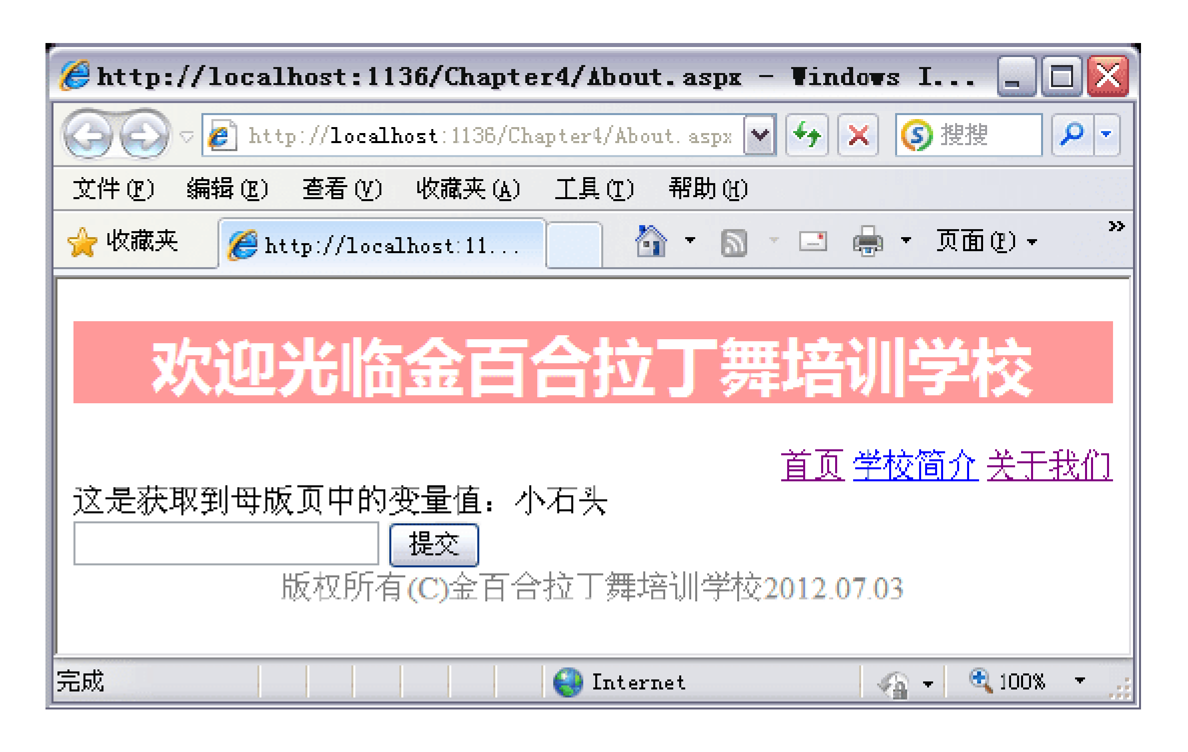


图 4-29　获取母版页中的变量

(10) 在文本框中输入新的变量值，然后单击“提交”按钮，之后的效果如图 4-30 所示。



图 4-30　为母版页中的变量设置新值

2. 动态访问母版页

在有些情况下，可能需要动态更改母版页。也就是通过编程动态地设置内容页的母版页。例如，可能希望允许用户从几个布局中进行选择，根据个人喜好来设置不同的母版页。

例 4-15：创建多个母版页，使用户可以选择不同的母版页。

(1) 启动 VWD 2010，打开网站 Chapter4。

(2) 通过“添加新项”对话框，添加一个名为 MasterPage2.master 的母版页，新创建的母版页和第 1 个母版页 MasterPage.master 非常相似，只是将标题栏和版权信息中的“金百合”换成“小石头”，然后将导航菜单左对齐，相应的代码如下所示：

```
    <form id="form1" runat="server">
      <div id="PageWrapper">
        <div id="top" align="center"><h1>欢迎光临小石头拉丁舞培训学校</h1></div>
        <div id="menu" align="left">
            <asp:ContentPlaceHolder id="menuContent" runat="server">
                <a href="Index.aspx">首页</a> <a href="Info.aspx">学校简介</a> <a
href="About.aspx">关于我们</a>
            </asp:ContentPlaceHolder>
        </div>
        <div id="main">
```

```
                <asp:ContentPlaceHolder ID="mainContent" runat="server">
                </asp:ContentPlaceHolder>
              </div>
              <div id="footer" align="center" style="color:Gray">版权所有(C)小石头拉丁舞培训学校
2012.07.03</div>
            </div>
        </form>
```

(3) 打开 Info.aspx 页面，设置 menuContent 控件“默认为母版页的内容”，在 mainContent 控件中添加一个 Label 控件和一个 DropDownList 控件。设置 Label 控件的 Text 属性为“选择母版页”；设置 DropDownList 控件的 AutoPostBack 属性为 True，为 DropDownList 添加两个选项，分别对应两个母版页，生成的代码如下：

```
    <asp:Content ID="Content3" ContentPlaceHolderID="mainContent" Runat="Server">
        <asp:Label ID="Label1" runat="server" Text="选择母版页"></asp:Label>
        <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
            onselectedindexchanged="DropDownList1_SelectedIndexChanged" >
            <asp:ListItem Value="MasterPage.master">金百合</asp:ListItem>
            <asp:ListItem Value="MasterPage2.master">小石头</asp:ListItem>
        </asp:DropDownList>
    </asp:Content>
```

(4) 为 DropDownList 控件添加 SelectIndexChanged 事件处理函数，代码如下：

```
        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
        {
            Session["masterpage"] = DropDownList1.SelectedValue;
            Response.Redirect(Request.Url.ToString());
        }
```

此代码将根据用户从 DropDownList 控件中的选中，将相应的母版页的文件名加载到一个持久的会话变量中，然后刷新当前页。

(5) 为 Info.aspx 页添加 PreInit 事件处理程序，将当前页的 MasterPageFile 属性设置为会话变量中的值(如果有)。此代码必须在 Page_PreInit 处理程序中运行，因为必须建立母版页，使得页面可以创建其实例，然后进一步初始化。代码如下：

```
        protected void Page_PreInit(Object sender, EventArgs e)
        {
            if (Session["masterpage"] != null)
            {
                this.MasterPageFile = (String)Session["masterpage"];
            }
        }
```

(6) 在 Info.aspx 页的 PreRende 事件中初始化 DropDownList 控件，根据会话变量中的

值(如果有)判断哪个选项被选中。代码如下：

```
protected void Page_PreRender(object sender, EventArgs e)
{
    if (Session["masterpage"] != null)
    {
        ListItemCollection items = DropDownList1.Items;
        for (int i = 0; i < items.Count; i++)
        {
            if (items[i].Value.Equals((String)Session["masterpage"]))
                items[i].Selected = true;
            else
                items[i].Selected = false;
        }
    }
}
```

(7) 测试 Info.aspx 页，默认将加载母版页 MasterPage.master，如图 4-31 所示。

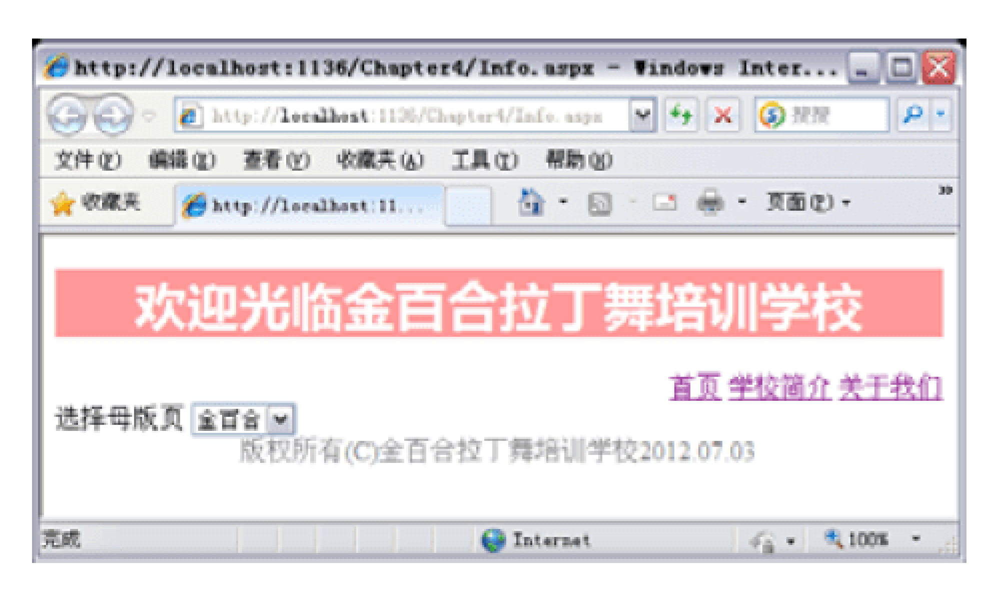


图 4-31　默认加载母版页 MasterPage.master

(8) 从下拉列表中选择“小石头”将会重新显示此页，但这一次使用的是 MasterPage2.master 母版页，如图 4-32 所示。

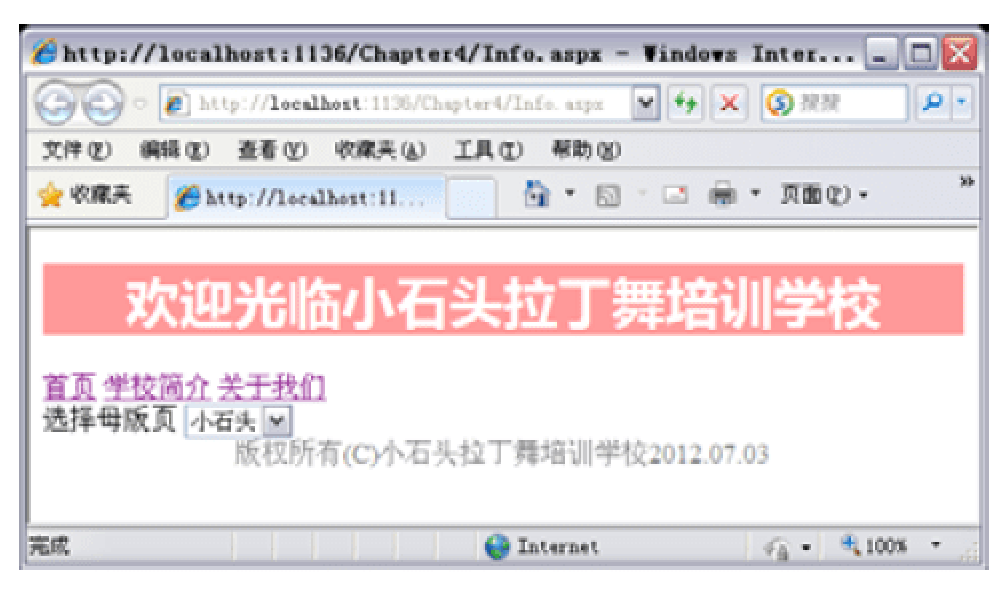


图 4-32　动态切换母版页

4.6　本章小结

站点中所有页面保持一致的外观有利于使站点看起来更专业和有吸引力。这样还有助

于访问者在站点中找到正确的信息，增加他们再次访问站点的可能性。ASP.NET 4 提供了大量帮助创建外观一致的 Web 站点的工具。本章首先介绍了 CSS 样式和页面布局技术，包括 CSS 的基本语法和样式规则以及 VWD 中与 CSS 相关的各种面板和工具的使用。接下来，学习了主题的定义与使用，主题用于改变站点中页面的外观以及它们所包含的控件。主题包含 CSS 文件、图像和外观，因此可以直接通过应用主题来改变页面的颜色、字体、位置和图像。最后，介绍了母版页的创建与使用技巧，ASP.NET 的母版页和内容页用于帮助创建布局，可以在每个基于该母版页的页面中重复使用这个布局。

4.7　思考和练习

1. VWD 提供了哪些使用 CSS 的便利工具？
2. 在下面两个规则中，哪个规则比较容易在 Web 站点中跨页面重用？请解释原因。

```
#MainContent
{
  border: 1px solid blue;
}
.BoxWithBorders
{
  border: 1px solid blue;
}
```

3. 如何定义关联选择符？
4. 设置页面的 Theme 属性与 StyleSheetTheme 属性，两者之间有何区别？
5. 当控件的属性和主题中控件的外观定义发生冲突时，哪个有较高优先级？
6. 如何将内容页中的 Content 控件与母版页中的 ContentPlaceHolder 关联起来？
7. 如何禁用主题？
8. 创建一个 CSS 规则，将站点中所有的一级标题(h1)的外观设置如下：
- 字体使用 Arial，并且加粗；
- 颜色为红色；
- 字体大小为 18 像素；
- 上边框和左边框为蓝色细边。

9. 设想用下面的外观定义创建了一个应用到站点中所有按钮的外观：

```
<asp:Button runat="server" CssClass="MyButton" />
```

该 CSS 类 MyButton 设置按钮的背景色为黑色，前景色为白色。为了使页面中的一个特定按钮引人注目，决定给它设置一个红色背景。有哪些方法可以控制这个按钮的外观？

10. 指出设置页面 Theme 属性的 3 种不同方式，并解释这些方式之间的区别。

第5章　数据访问与数据绑定

ASP.NET 应用程序的数据访问是通过 ADO.NET 进行的，ADO.NET 可以使 Web 应用程序从各种数据源中快速访问数据。本章首先介绍数据库的基本知识和 SQL 语言，接着介绍 ADO.NET 访问 SQL Server 数据库和访问 XML 数据的方法，最后介绍了 ASP.NET 提供的数据绑定技术和数据控件的使用。通过本章的学习，读者应该掌握如何在 ASPX 页面中访问和操作数据源，以及数据信息的显示与更新。学会设计主-从页面显示数据信息。

本章学习目标：

- 新建数据库和表
- 使用 SQL 来操纵数据
- ADO.NET 的基本知识
- ADO.NET 访问数据库的方法
- 掌握单值和列表控件的数据绑定
- 理解数据源控件的工作原理
- 掌握 GridView 控件的使用方法和技巧
- 学会设计主-从页面显示数据
- ADO.NET 读写 XML 文件

5.1　数据库基础知识

数据库是非常有用的，因为它允许通过结构化的方式来存储和检索数据。数据库最大的好处是能够在运行时被访问，这就意味着在 VWD 中，可以使用数据库存储评论、音乐流派、图片、用户信息(用户名、电子邮件地址、密码等)和日志信息等，然后可以通过 ASPX 页面访问这些数据。

5.1.1　什么是数据库

简单来说，数据库就是以易于访问、管理和更新的形式排列的数据的集合。例如，日常生活中，用笔记本记录亲戚和朋友的联系方式，将他们的姓名、地址以及电话等信息都记录下来。这个“通讯录”就是一个最简单的“数据库”，每个人的姓名、地址以及电话等信息就是这个数据库中的“数据”。在计算机领域，数据库是指长期存储在计算机内的、有组织的、可共享的、统一管理的相关数据的集合。

最为流行的一种数据库是关系数据库(relational database)。这种数据库常用于 Web 站

点中，也将用于本书后续部分。不过，关系数据库并不是唯一的数据库类型。还有其他类型的数据库，包括平面文件数据库、对象关系数据库和面向对象数据库，但它们在 Internet 应用程序中不常见。

关系数据库中有表(table)的概念，其中数据以行和列的形式存储，如同电子表格一样。表中的每行包含存储于其中的记录项的完整信息，而每列包含表中记录项的特定属性的信息。

“关系”指的是数据库中不同表相互关联的方式。它不是将相同的数据一遍遍地复制，而是在其自己的表中存储重复的数据，然后从其他表中创建与该数据的关系。

在 ASP.NET 项目中，可以使用多种不同类型的数据库，包括 Microsoft Access、SQL Server、Oracle 和 MySQL。不过，在 ASP.NET 4 Web 站点中最常用的数据库是 Microsoft SQL Server。本书主要使用 Microsoft SQL Server 2008 Express Edition，因为它是随 VWD 免费提供的，有着许多创新性的功能。而且，由于其数据库引擎与 SQL Server 2008 商业版相同，因而可以在开发周期的后续阶段轻松地升级到商业版本。

5.1.2　新建数据库和表

要使用数据库首先就要创建数据库和表。本节将介绍如何新建 SQL Server 数据库，并在数据库中新建数据表和表之间的关系。本书后面的章节将主要使用该数据库。

1. 新建 SQL Server 数据库

例 5-1：在 VWD 的“服务器资源管理器”窗口中连接到 SQL Server 服务器，并新建数据库 WeiBo。

(1) 首先，选择“视图”|“服务器资源管理器”命令，打开“服务器资源管理器”窗口。

(2) 在“服务器资源管理器”窗口中右击“数据连接”，在弹出的快捷菜单中选择“新建 SQL Server 数据库”命令，如图 5-1 所示。

(3) 此时将打开“创建新的 SQL Server 数据库”对话框，如图 5-2 所示。

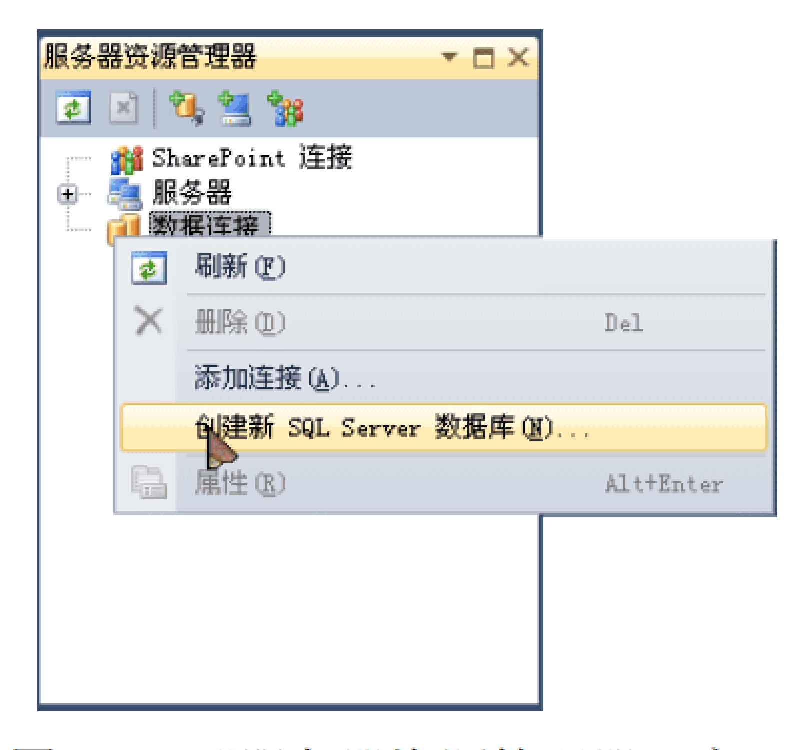


图 5-1　“服务器资源管理器”窗口



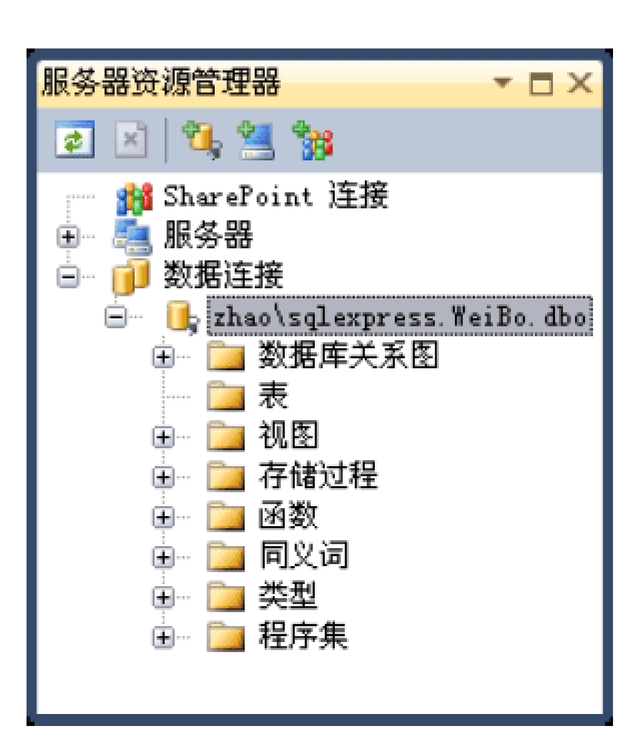
图 5-2　“创建新的 SQL Server 数据库”对话框

(4) 在“服务器名”文本框中输入 SQL Server 数据库服务器名称，通常为“[机器名]\[数据实例名]”的格式，然后选择登录到服务器的方式，最后在“新数据库名称”文本框中输入新建数据库的名称 WeiBo。

(5) 单击“确定”按钮，便创建了一个空数据库 WeiBo，此时可以在“服务器资源管理器”窗口中看到连接到 WeiBo 的数据连接，如图 5-3 所示。

说明：

如果要连接到已有的数据库，则可以在右击“数据连接”时从弹出的快捷菜单中选择“添加连接”命令，在打开的“添加连接”对话框中进行操作即可，如图 5-4 所示。



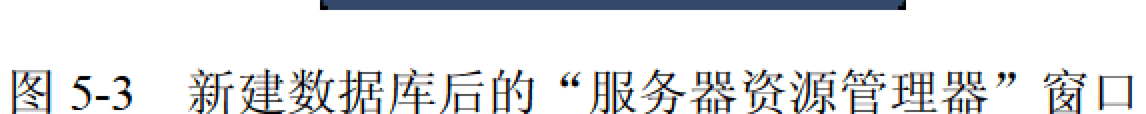

图 5-3　新建数据库后的“服务器资源管理器”窗口



图 5-4　“添加连接”对话框

有了与数据库的连接，就可以处理该数据库中的对象。使用 VWD 的内置数据库工具创建和操纵 SQL Server 2008 数据库表很容易。接下来就介绍如何在数据库中创建自己的表。

2. SQL Server 中的数据类型

与 Visual Basic .NET 和 C#这样的编程语言一样，SQL Server 数据库也使用不同的数据类型来存储数据。SQL Server 2008 支持 30 多种不同的数据类型，大部分与.NET 中使用的类型相似。表 5-1 列出了最常用的 SQL Server 数据类型及其说明。

表 5-1　SQL Server 数据类型

SQL Server 2008 数据类型	描　述	对应的.NET 数据类型
bit	以 0/1 格式存储布尔值(1 表示 True，0 表示 False)	System.Boolean
char/nchar	包含固定长度的文本。如果存储的文本短于定义的长度，就用空格填充。nchar 以 Unicode 格式存储数据，允许存储用各种语言编写的数据	System.String
datetime	存储日期和时间	System.DateTime
datetime2	与 datatime 类型相似，但具有更高的精度和范围	System.DateTime
date	存储日期，没有时间元素	System.DateTime
time	存储时间，没有日期元素	System.TimeSpan

(续表)

SQL Server 2008 数据类型	描　　述	对应的.NET 数据类型
decimal	允许存储较大的小数，最多可存储 38 个数字	System.Decimal
float	允许存储较大的小数，被称为近似的数据类型，近似数字数据类型并不存储为多数数字指定的精确值，它们只储存这些值的最近似值	System.Double
image	允许存储大的二进制对象，如文件。尽管其名称暗示了只可用它存储图像，但事实并非如此。可用它存储任何类型的文档或其他二进制对象	System.Byte[]
tinyint	用于存储 0~255 之间的整数	System.Byte
smallint	用于存储−32 768~32 767 之间的整数	System.Int16
int	用于存储−2 147 483 648~2 147 483 647 之间的整数	System.Int32
bigint	用于存储−9 223 372 036 854 775 808~9 223 372 036 854 775 807 之间的较大整数	System Int64
text/ntext	用于存储较多的文本	System.String
varchar/nvarchar	用于存储变长的文本。nvarchar 以 Unicode 格式存储数据，这样就可以存储用各种语言编写的数据	System.String
uniqueidentifier	存储全局唯一标识符	System.Guid

其中的一些数据类型允许指定最大长度。在定义 char、nchar、varchar 或 nvarchar 类型的列时，需要指定字符长度。例如，nvarchar(10)最多可存储 10 个字符。从 SQL Server 2005 开始，到 SQL Server 2008 之前的 SQL Server 版本，这些数据类型都允许指定 MAX 为最大值。通过 MAX 说明符，可以在单个列中最多存储 2GB 的数据。对于大段的文本，像评论主体部分，应该考虑使用 nvarchar(max)数据类型。如果清楚某列(像邮政编码或手机号)的最大长度或想显式限制其长度，则可以指定这一长度。例如，评论的标题应存储于 nvarchar(200)的列中，限制最大字符数为 200。

3. 主键和标识列

主键是为了唯一标识表中记录的一个或多个列。如果将一列标识为主键，那么数据库引擎就可以确保最终不会出现具有相同值的两个记录。主键可以由单个列(例如，包含了表中每条记录的唯一数值的数字列)组成，也可以由多个列组成，这些列组合起来构成整条记录的唯一 ID。

SQL Server 也支持标识列。标识列是一个数字列，其值是在插入新记录时自动生成的。它们通常用作表的主键。

说明：

主键不是必须的，但是设置了主键可以使数据库编程人员的工作变得简单，因此建议为表添加主键。

4. 设计并创建表

前面已经创建了数据库 WeiBo，该数据库将设计为一个简易微博系统的后台数据库。微博是随着 Web 2.0 而兴起的一类开放的互联网社交服务，因其具有简单易用、门槛低、传播即时等特点而迅速吸引了众多互联网爱好者。它允许用户以简短文字随时随地更新自己的状态，每条信息的长度都在 140 字以内，每个用户既是微内容的创造者也是微内容的传播者和分享者。

例 5-2：设计 WeiBo 数据库中的表，并使用 VWD 的内置数据库工具创建这些表。

(1) 在“服务器资源管理器”窗口中，展开“数据连接”，然后继续展开例 5-1 中创建的数据库 WeiBo。

(2) 右击下面的“表”节点，从弹出的快捷菜单中选择“添加新表”命令，将打开表的“设计”窗口，如图 5-5 所示。在此窗口中，可输入构成表定义的列名和数据类型。

(3) 首先来设计用户信息表 Z_USER。该表主要是存储用户的相关信息，具体字段信息如表 5-2 所示。

表 5-2　Z_USER 表字段信息

字 段 名 称	数 据 类 型	描　　述
user_id	int	用户编号，主键，标识列
user_name	nvarchar(50)	用户名，非空
user_login	varchar(50)	登录名，非空
user_password	varchar(20)	登录密码，非空
user_sex	varchar(2)	性别，只能为“男”或“女”，非空
user_photo	image	用户头像，可空
user_email	varchar(32)	用户邮箱，email，可空
regist_time	datetime	注册时间，非空
user_address	nvarchar(64)	用户地址，可空
user_birthday	datetime	出生日期，可空
user_telephone	varchar(16)	手机号码，可空
home_url	varchar(20)	微博地址，非空，其他用户访问时所用的 url
user_info	nvarchar(64)	自我介绍，可空

输入 user_id 列并选择数据类型后，取消选择“允许为 null”列的复选框，即设置该列非空。接着，单击该列单元格左侧的黑色箭头标识，选取整行，然后单击“表设计器”工具栏中的“设置主键”按钮，如图 5-6 所示，将该列设置为主键。

说明：

默认情况下，标识列的第一条记录的值为 1，后续记录依次加 1。可以通过“标识规范”选项中的“标识增量”和“标识种子”进行设置，来改变列的默认行为。

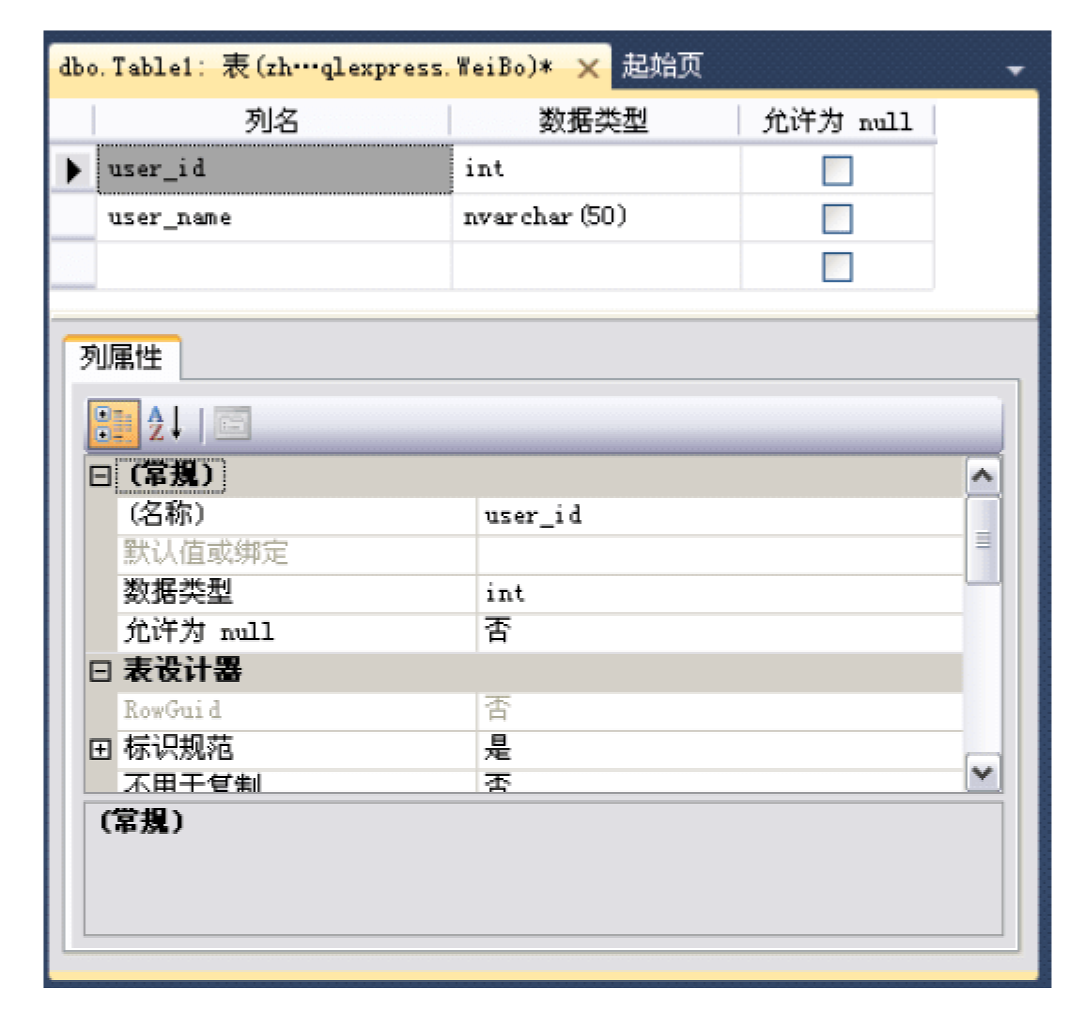


图 5-5　表的“设计”窗口

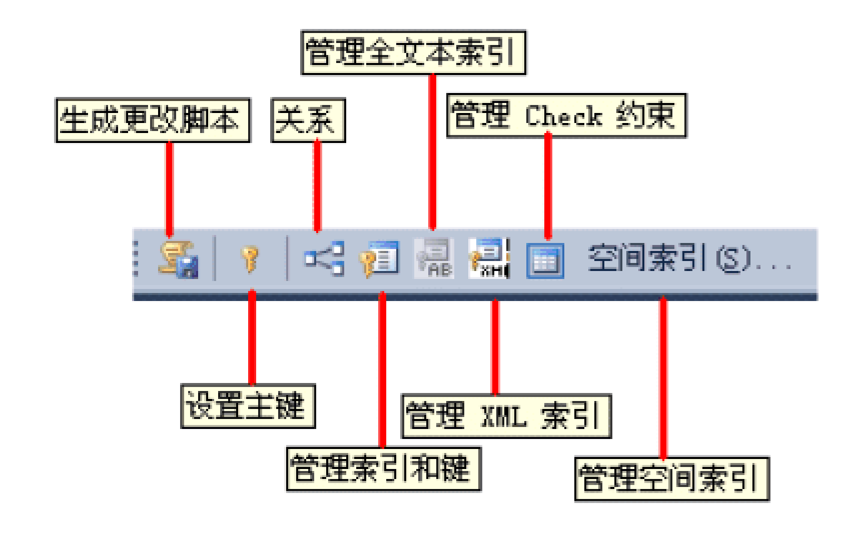


图 5-6　“表设计器”工具栏

在表定义的下面是“列属性”面板，在 user_id 列仍处于选中状态下时，展开“列属性”面板中的“标识规范”选项，将“(是标识)”选项设置为“是”，如图 5-7 所示，将该列设置为标识列。

按表 5-2 所示设置完其他字段信息后，单击工具栏中的“保存”按钮，将弹出“选择名称”对话框，要求为该表提供一个名称，如图 5-8 所示。输入 Z_USER 作为表名，然后单击“确定”按钮，完成表 Z_USER 的创建。



图 5-7　设置“标识规范”属性



图 5-8　“选择名称”对话框

(4) 接下来设计信息表 Z_MESSAGE，该表存储用户发表的微博信息，具体字段信息如表 5-3 所示。

表 5-3　Z_MESSAGE 表字段信息

字 段 名 称	数 据 类 型	描　　述
msg_id	int	信息编号，主键，标识列
user_id	int	用户编号，非空
msg_content	nvarchar(140)	信息内容，非空
reply_count	int	评论次数，非空
post_time	datetime	发表时间，非空

(5) 消息评论表 Z_REPLY 用于存储其他用户评论某条信息的记录，具体字段信息如表 5-4 所示。

表 5-4　Z_ REPLY 表字段信息

字 段 名 称	数 据 类 型	描　　述
reply_id	int	评论编号，主键，标识列
msg_id	int	信息编号，非空
reply_user_id	int	评论用户编号，非空
src_user_id	int	被评论用户编号，非空
reply_content	nvarchar(140)	评论内容，可空
reply_time	datetime	评论时间，非空

(6) 不同用户之间可以收听，所以还需要一个关注用户表 Z_USER_FUN，具体的字段信息如表 5-5 所示。

表 5-5　Z_ USER_FUN 表字段信息

字 段 名 称	数 据 类 型	描　　述
fun_id	Int	收听用户关系编号，主键，标识列
user_id	Int	用户编号，非空
fun_user_id	Int	收听用户编号，非空

技巧：

一旦创建好表，如果要对其进行修改，如增加字段或修改某个字段的数据类型等，可以右击表名，从弹出的快捷菜单中选择“打开表定义”命令，打开表的设计视图进行修改。

也可以给上述表中的时间类型的字段列指定默认值。默认值是在插入数据时，如果没指定该列的数据，数据库会自动插入默认值。对于上述时间类型的列，可以设置默认值为 getdate()，它会自动插入当前日期和时间。

5. 创建表之间的关系

在前面创建的 4 张表中，很显然，表之间存在一定的关系。例如，信息表 Z_MESSAGE 中的 user_id 应该来源于 Z_USER 表中的 user_id，以标识该信息是哪个用户发表的。

现在，假设有一个用户注销了微博账户，需要删除 Z_USER 表中相应的记录。如果没有关系存在，那么数据库会允许进行这一操作。不过，这会带来一些麻烦。因为当再次尝试显示 Z_MESSAGE 表中该用户发表的信息时，将无法获取该信息作者的详细资料。

为了避免这类问题，使数据库保持良好的、一致的状态，可以在两表之间创建关系。在建立了正确的关系后，数据库将阻止无意间从一个仍有其他记录与之关联的表中删除记录。

除了保护数据外，关系也使得数据模型更清晰。通过关系图查看数据库，有助于更好

地理解表的连接方式和它们表示的数据。

可以通过在一个表的主键和另一个表的列之间创建关系来进行关系的定义。在另一个表中的列通常被称为外键(foreign key)。

例 5-3：在表之间创建关系。

(1) 在“服务器资源管理器”窗口中，展开“数据连接”，然后继续展开例 5-1 中创建的数据库 WeiBo。

(2) 在表之间可视化地添加关系之前，需要为数据库添加一个关系图。关系图是一个可帮助理解和定义数据库的可视化工具。在关系图上，可以将表的一列拖至另一个表中来创建关系。右击“数据库关系图”节点，从弹出的快捷菜单中选择“添加新关系图”命令。如果是首次将关系图添加到数据库中，将会出现一个对话框，询问是否希望 VWD 将执行该操作的用户作为数据库的所有者。单击“是”按钮继续。

(3) 在随后打开的“添加表”对话框中，选择前面创建的所有表，单击“添加”按钮将表添加到关系图中，如图 5-9 所示。然后单击“关闭”按钮关闭对话框。

(4) 在 Z_USER 表上，单击 user_id 列(它应包含黄色钥匙图标，表明这是表的主键)的左侧，然后将它拖至 Z_MESSAGE 表的 user_id 列上并释放鼠标键。此时将弹出“表和列”对话框，指定了新建关系的名称和主键表与外键表中的字段列，如图 5-10 所示。

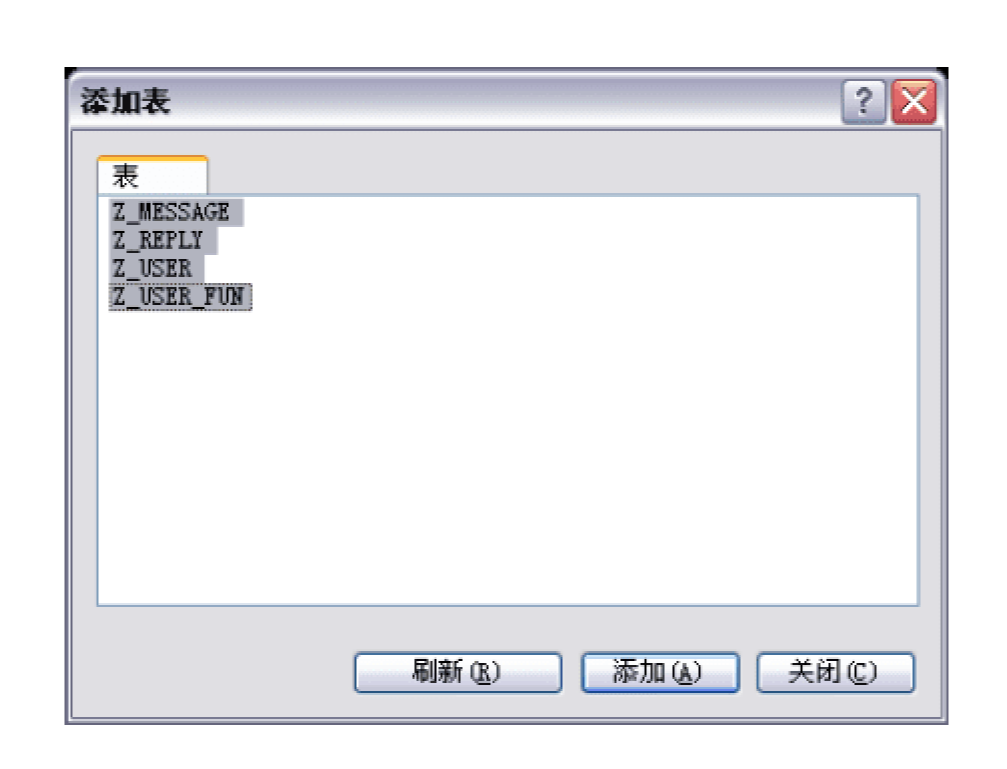


图 5-9　“添加表”对话框

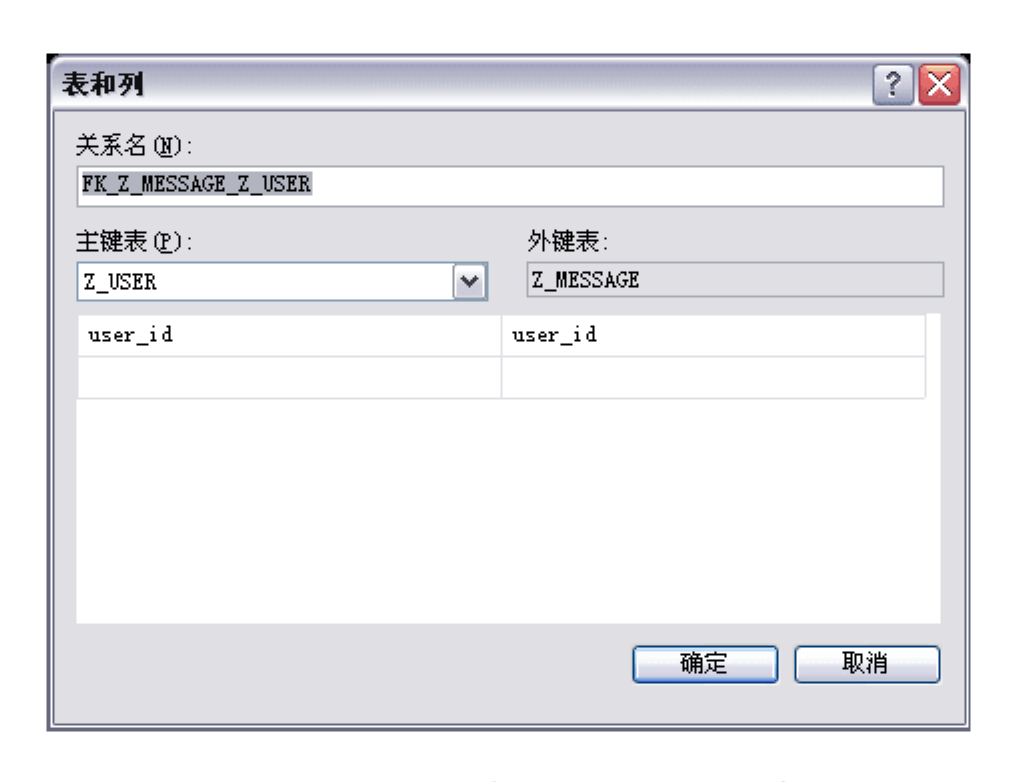


图 5-10　“表和列”对话框

(5) 单击“确定”按钮关闭“表和列”对话框，将显示如图 5-11 所示的“外键关系”对话框，将“强制外键约束”选项设置为“是”。这一属性确保了如果还有信息与 Z_USER 表中相应的 user_id 的记录关联，那么就不能从该 Z_USER 表中删除记录。

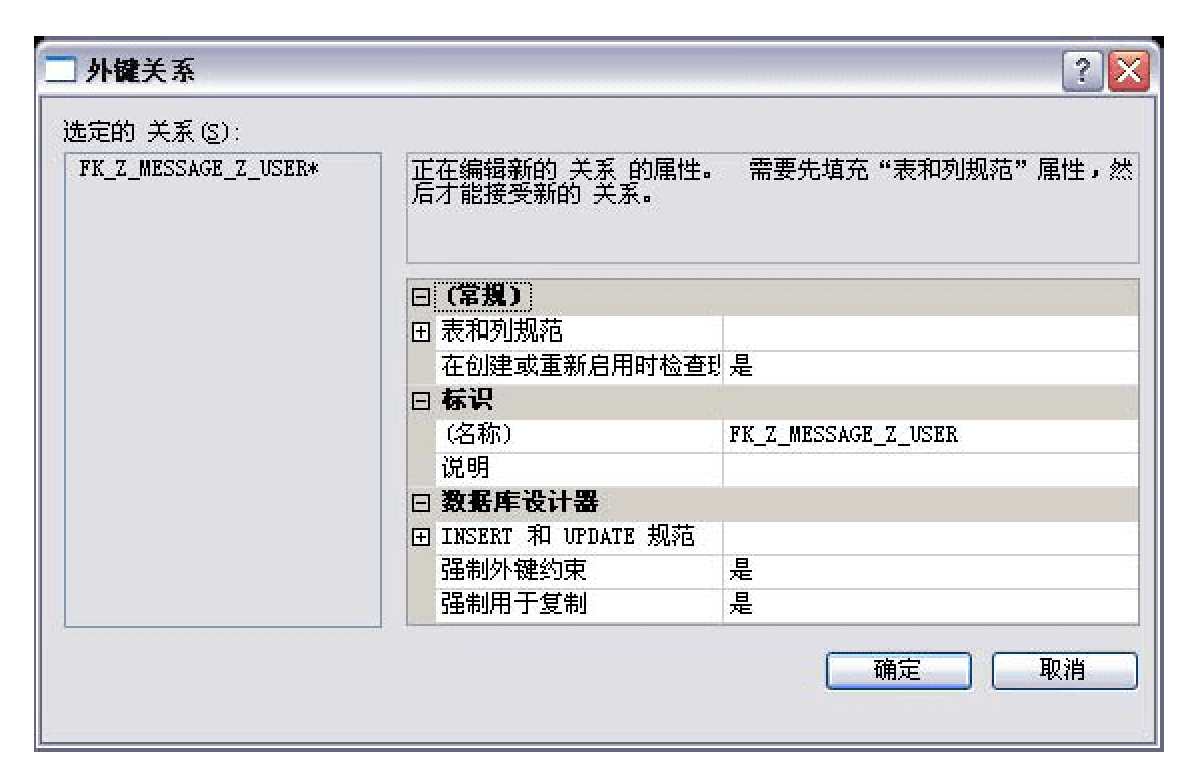


图 5-11　“外键关系”对话框

(6) 单击“确定”按钮完成关系的创建，此时，关系图中两表之间显示一条线。在 Z_USER 表这一边可以看到用一个黄色钥匙图标表明该表包含关系的主键。而在另一边可以看到用无穷符号(∞)表明 Z_MESSAGE 表中有多条使用相同 user_id 的记录，如图 5-12 所示。

注意：

这两个表之间的线并不一定总是指向正确的列。这样很容易使人混淆，为了确认参与关系中的列，可右击两个表之间的线，从弹出的快捷菜单中选择“属性”命令，在“属性”面板中找到“表和列规范”选项，即可查看哪个列和表位于关系中，如图 5-13 所示。

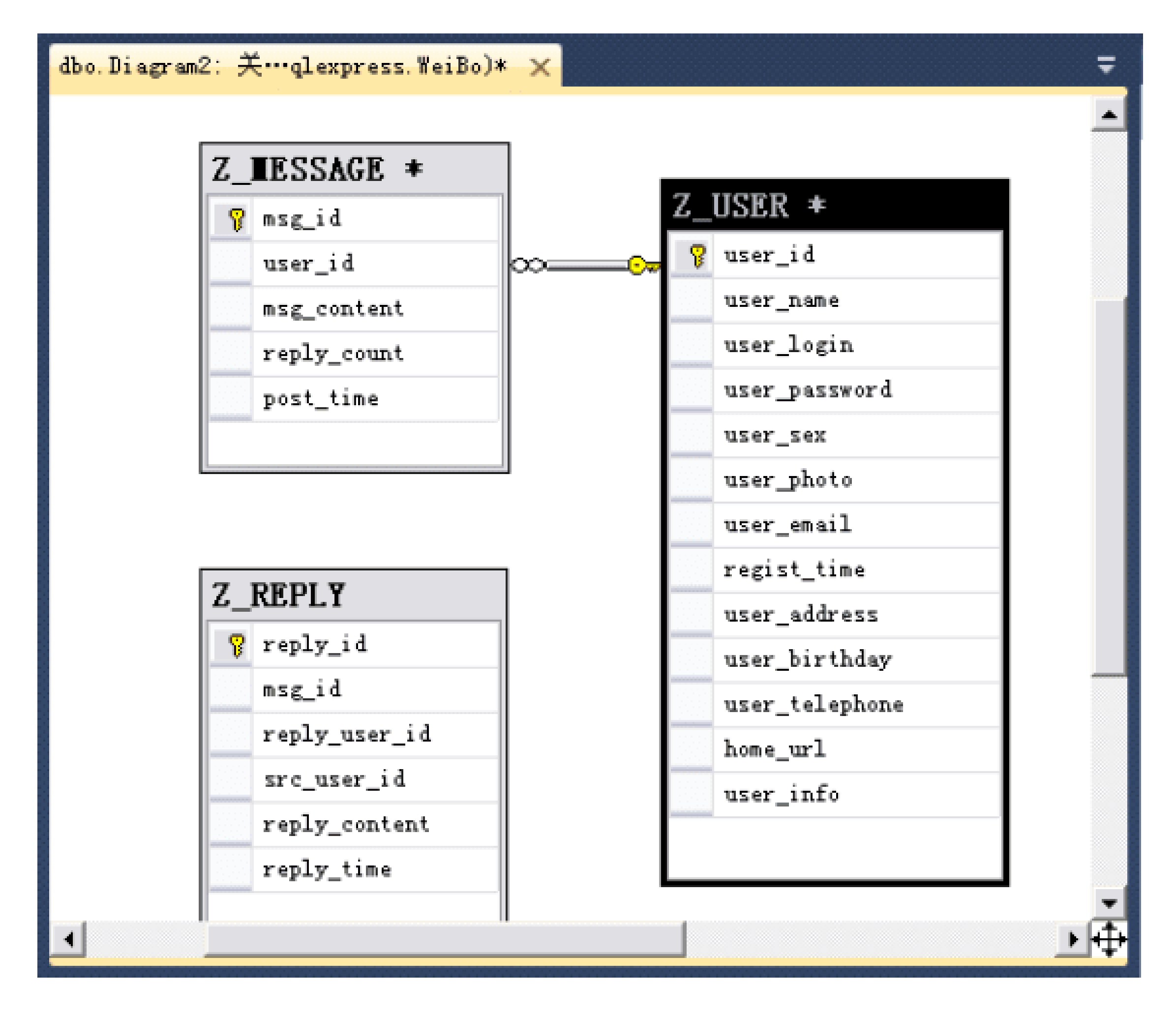


图 5-12　关系图显示效果

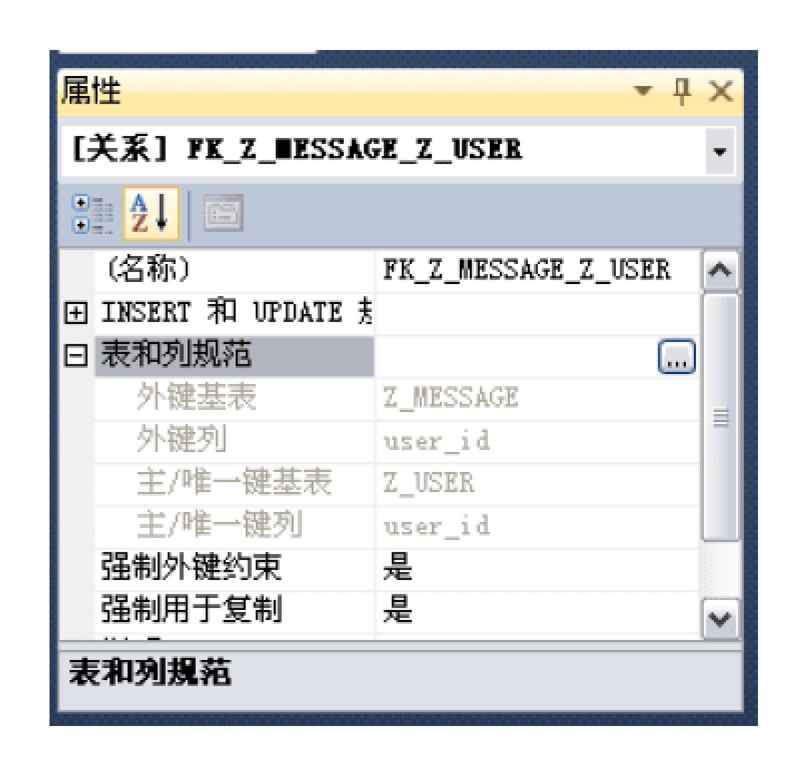


图 5-13　查看“表和列规范”属性

(7) 用同样的方法建立其他表之间的关系：Z_REPLAY 表的 msg_id 与 Z_MESSAGE 表中的 msg_id 关联；Z_REPLAY 表的 reply_user_id、src_user_id 均与 Z_USER 表的 user_id 关联；Z_USER_FUN 表中的 user_id、fun_user_id 均与 Z_USER 表的 user_id 关联。

(8) 单击“保存”按钮，保存对关系图的更改。可以将名称设置为其默认值 Diagram1 或是输入一个更具描述性的名称。

6. 添加和删除表中的数据

如果在两个表之间创建了关系，那么在试图插入、修改或删除数据时，数据库将强制这一关系。这里将通过 VWD 内置的数据库工具向表中插入数据，然后进行删除操作，验证表间关系的约束。后面的章节将会介绍如何在自己的 Web 站点上处理这种情况，以及如何向用户显示友好的错误提示。

例 5-4：通过 VWD 内置的数据库工具，向表中插入一些数据，然后尝试删除具有外键关系的主表中的记录，验证外键关系的约束。

(1) 在“服务器资源管理器”窗口中，右击 Z_USER 表，从弹出的快捷菜单中选择“显示表数据”命令，打开表的数据视图。

(2) 在除 user_id 之外的其他非空字段中分别输入一些用户信息，然后按 Tab 键离开当前行，则该行数据就被插入到数据库中了，并且 user_id 列会自动用一个唯一的序号填充。

最终得到的列表如图 5-14 所示。

user_id	user_name	user_login	user_password	user_sex	user_...	user_email	regist_time	user_address	user_birthday	user_tele...	home_url	user_info
1	赵艳铎	yanduozh	zhao1234	男	NULL	zhaoyanduo@t...	2012-7-7 10:28:59	北京市海淀区	1981-4-29 11...	15910806516	yanduozh	我自豪我骄
2	小石头	xst	12345stone	男	NULL	littlestone@...	2012-7-7 10:31:42	浙江省杭州市	2010-3-19 22...	18801012313	xst875614341	我妈妈跟人
3	葛萌萌	dahai	dahai123	女	NULL	gemm@qq.com	2012-7-7 10:33:17	河北省沧州市	1986-1-10 16...	13831705888	gemm	永远的大海
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 5-14　Z_USER 表中插入的记录

(3) 用同样的方法打开 Z_USER_FUN 表，向其中的 user_id 和 fun_user_id 列输入数据，由于这两列都是与 Z_USER 表中的 user_id 关联的，所以只能输入 Z_USER 表中存在的值(当前只能是 1、2、3)，如果试图输入其他值，将会弹出如图 5-15 所示的提示对话框。

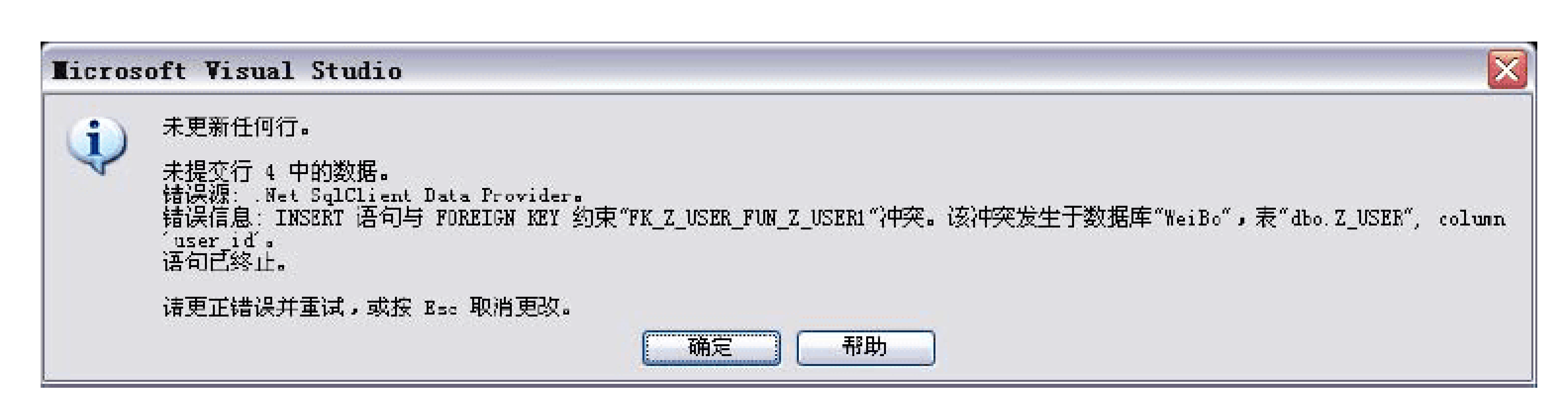


图 5-15　插入数据时外键约束冲突提示对话框

(4) 在 Z_USER_FUN 表输入一些数据后，再次打开 Z_USER 表的“数据”视图，选中其中某条记录，单击鼠标右键，从弹出的快捷菜单中选择“删除”命令，VWD 将显示如图 5-16 所示的提示对话框，提示不能删除该记录。

说明：

只有当要删除的用户对应的 user_id 在所有外键关联表中都没有记录时，才能删除该用户。

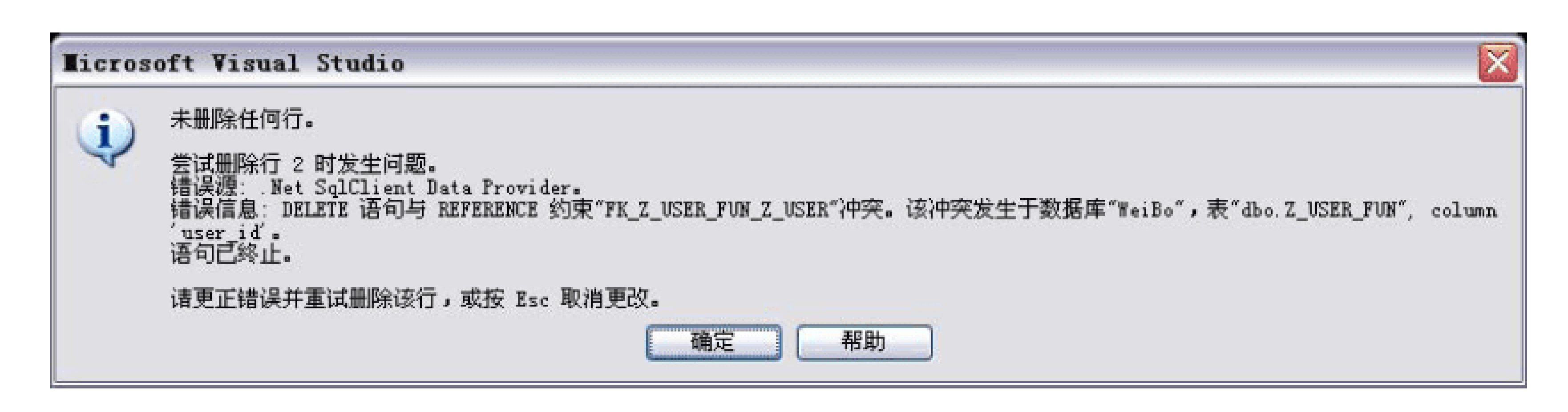


图 5-16　删除数据时外键约束冲突提示对话框

5.2　SQL 简介

为了成功地在 ASPX 页面中运用数据库，将学习如何使用一种名为 SQL(Structured Query Language，结构化查询语言)的查询语言来访问数据库。它是用于查询关系数据库的实际语言，几乎所有的关系数据库系统都能理解它。SQL 有许多明确的标准，其中最流行的是 ANSI 92 SQL 标准。

5.2.1　SQL 概述

SQL 语言是一种介于关系代数和关系演算之间的结构化查询语言，其功能并不仅仅是查询，还具备数据定义和数据操纵等功能。ANSI(美国国家标准协会)规定 SQL 为关系型数据库管理系统的标准语言。目前，绝大多数流行的关系型数据库管理系统，如 Oracle、Sybase、Microsoft SQL Server、Access 等，都采用了 SQL 语言标准。

Microsoft SQL Server 2008 数据库支持 ANSI 92 SQL 标准中定义的大部分语法。在这一标准之上，Microsoft 又添加了一些专用扩充，合起来称为 T-SQL(Transact SQL)。本书后续部分都会使用 SQL 这一术语。

总的来说，SQL 语言具有如下特点。

- 类似于英语自然语言，简单易学。
- 是一种非过程语言。
- 是一种面向集合的语言。
- 既可独立使用，又可嵌入到宿主语言中使用。
- 具有查询、操纵、定义和控制一体化功能。

SQL 语言包含 4 个部分。

- 数据定义语言(DQL，Data Definition Language)：CREATE、ALTER、DROP
- 数据查询语言(DQL，Data Query Language)：SELECT。
- 数据操纵语言(DQL，Data Manipulation Language)：INSERT、UPDATE、DELETE
- 数据控制语言(DQL，Data Control Language)：COMMIT、ROLLBACK

对于数据库编程人员来说，与数据库交互时，使用最多的就是查询和操纵数据。所以本书重点介绍 SELECT、INSERT、UPDATE、DELETE 4 个语句。

说明：

SQL 语言不区分大小写，SELECT 与 Select 的含义是相同的。为了统一起见，本书中所书写的 SQL 命令全部使用大写形式。

5.2.2　SELECT 语句

要从数据库中读取数据，首先需要指定要从哪个表中检索什么列，这是通过 SELECT 语句完成的；需要使用 FROM 关键字指定表名；然后需要筛选数据，确保只返回符合条件的记录，可以使用 SQL 语句中的 WHERE 子句筛选数据；最后，可以使用 ORDER BY 子句对结果进行排序。

完整的 SELECT 语句的语法格式如下，其中方括号中的子句都是可选的：

```
SELECT 目标表的列名或列表达式集合
FROM 基本表或(和)视图集合
［WHERE 条件表达式］
```

```
［GROUP BY 列名集合 ［HAVING 组条件表达式］］
［ORDER BY 列名［集合］…］
```

首先从 FROM 子句列出的表中，选择满足 WHERE 子句给出的条件表达式的记录，然后按 GROUPBY 子句(分组子句)中指定列的值进行分组，再检索出满足 HAVING 子句中组条件表达式的组，按 SELECT 子句后面的列名或列表达式输出。ORDER 子句(排序子句)用来对输出的记录进行排序。

SELECT 语句既可以完成简单的单表查询，也可以完成复杂的连接查询和嵌套查询。下面以 WeiBo 数据库中的表为例介绍 SELECT 语句的各种用法。

1. 简单的 SELECT 语句

简单的 SELECT 语句格式如下。

```
SELECT 目标表的列名或列表达式集合
FROM 基本表或(和)视图集合
```

例如：

```
SELECT user_name, user_telephone, user_sex
FROM Z_USER
```

这个 SELECT 语句将返回用户表中的选定字段(user_name,user_telphone,user_sex)的数据。如果需要返回用户表中的所有字段，则可以使用星号(*)来代替所有列名，如下语句：

```
SELECT *
FROM Z_USER
```

使用 TOP 关键字可以只显示一组记录中前面的几个记录，它紧跟在 SELECT 关键字的后面，下面的例子中将返回用户表 Z_USER 的前 10 条记录。

```
SELECT TOP 10 *
FROM Z_USER
```

2. 消除重复行

两个本来并不完全相同的元组，投影到指定的某些列后，可能变成相同的行了，这时可以使用 DISTINCT 关键字来消除重复的行。

例如，以下 SQL 语句查询 Z_USER 表中的 user_sex 列：

```
SELECT user_sexFROM Z_USER;
```

由于用户的性别只有“男”或“女”两种取值，所以，上述的 SQL 语句执行后将会包含许多重复的行，如果想消除重复的行，可以通过指定 DISTINCT 关键字来实现：

```
SELECT DISTINCT user_sexFROM Z_USER;
```

如果没有指定 DISTINCT 关键字，则默认为 ALL，即保留结果表中所有重复的行。

3. 重命名输出列

在创建表时，为方便计算和查询，一般将字段名称定义为英文名。但是，当需要将表显示出来时，对于中国人来说，英文字段名称就不如中文字段名称直观。SQL 提供了 AS 关键字来对字段重新命名。例如，以下将输出的 user_name 字段重新命名为“姓名”，将 user_sex 字段重新命名为“性别”：

```
SELECT user_name AS 姓名, user_sex AS 性别
FROM Z_USER
```

注意：

在 SQL 查询语句中使用 AS 对表的字段重新命名，只改输出，并未改变该表在设计视图中的字段名称。

AS 关键字不仅可以对存在的字段命名，也可以对不存在的字段重命名，一般在以下情况中使用。

(1) 所涉及的表的字段名很长或者想把英文字段名改为中文字段名，使字段在结果集中更容易查看和处理。

(2) 查询产生了某些计算字段、合并字段等原本不存在的字段，需要命名。

(3) 多表查询中在两个或者多个表中存在重复的字段名。

另外，在 SELECT 语句中数字类型的字段之间还支持加、减、乘和除(+ − ×/)基本算术操作。而字符型字段之间也支持加操作，其结果是将两个字符串合并在一起，下面的语句演示了如何把用户表中的 user_name 和 user_address 连接起来：

```
SELECT user_name + user_address AS 用户信息
FROM Z_USER
```

4. 使用合计函数

为了进一步为用户提供方便的操作途径，增强检索功能，SQL 提供了许多合计函数，也称为聚集函数。主要的合计函数如表 5-6 所示。

表 5-6　SQL 提供的合计函数

合 计 函 数	功　能
COUNT([DISTINCT|ALL]*)	统计元组个数
COUNT([DISTINCT|ALL]<列名>)	统计一列中值的个数
SUM([DISTINCT|ALL]<列名>)	计算一列值的总和(此列必须是数值型)
AVG([DISTINCT|ALL]<列名>)	计算一列值的平均值(此列必须是数值型)
MAX([DISTINCT|ALL]<列名>)	求一列值中的最大值
MIN([DISTINCT|ALL]<列名>)	求一列值中的最小值

如果指定 DISTINCT 关键字，则表示在计算时要取消指定列中的重复值。如果不指定

DISTINCT 关键字或指定 ALL 关键字(ALL 为默认值)，则表示不取消重复值。

以下 SQL 语句查询 Z_USER 表中的总用户数：

```
SELECT COUNT(*)
FROM Z_USER;
```

说明：

在合计函数遇到空值时，除 Count(*)外，都跳过空值而只处理非空值。

5. 使用 WHERE 子句筛选数据

在同样的检索情况下，使用 WHERE 子句可以指定查询条件，缩小数据检索范围，例如，要在用户表 Z_USER 中检索出姓名 user_name='葛萌萌'的记录，可以使用如下 SQL 语句：

```
SELECT *
FROM    Z_USER
WHERE    user_name = '葛萌萌'
```

说明：

“葛萌萌”被两个单引号括了起来，是为了表示该值是文本(字符串)类型。

当 WHERE 子句的约束条件比较多或者比较复杂时，可以在 WHERE 子句中使用一些比较运行符和逻辑运算符，如表 5-7 所示。

表 5-7　常用的查询条件

查 询 条 件	谓　词
比较	=，>，<，>=，<=，!=，<>，!>，!<，Not+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件(逻辑运算)	AND，OR，NOT

LIKE 操作符是针对字符串的，作用是确定给定的字符串是否与指定的模式匹配。模式可以包含常规字符和通配符字符。模式匹配过程中，常规字符必须与字符串中指定的字符完全匹配。可以使用字符串的任意片段匹配通配符。与使用 = 和 != 字符串比较运算符相比，使用通配符可使 LIKE 运算符更加灵活。

例如，要查询 Z_USER 表中所有姓“赵”的男性用户，SQL 语句如下：

```
SELECT *
FROM Z_USER
WHERE user_name LIKE '赵%' AND user_sex= '男'
```

6. ORDER BY 子句

在用 WHERE 子句定义了筛选要求后，如果想改变从数据库返回的数据顺序，可以使用 ORDER BY 子句。ORDER BY 子句出现在 SQL 语句的末尾，可包含一个或多个列名或表达式，也可以包括 ASC 或 DESC 来决定记录是以升序(ASC，默认关键字)或以降序(使用 DESC)排列，如果没有指定排序常数，将默认以升序方式排列，其语法格式如下：

```
ORDER BY  字段或者是字段集合  排序常数
```

例如，将 Z_USER 表中所有性别为“女”的数据按姓名进行降序排列，SQL 语句如下：

```
SELECT *
FROM Z_USER
WHERE    user_sex = '女'
ORDER BY    user_name    DESC
```

7. GROUP BY 子句

GROUP BY 子句将查询结果按某一列或多列的值分组，值相等的为一组。

对查询结果分组的目的是为了细化聚集函数的作用对象。如果未对查询结果进行分组，聚集函数将作用于整个查询结果。分组后聚集函数将作用于每一个组，即每一个组都有一个函数值。

以下 SQL 语句将查询每个用户收听的用户总数：

```
SELECT user_id, COUNT(user_id) AS  收听用户数
FROM Z_USER_FUN
GROUP BY user_id
```

该语句对查询结果按 user_id 进行分组，所有具有相同 user_id 值的元组为一组，然后对每一组作用聚集函数 COUNT 计算，以求得该组的总数。

如果分组后还要求按一定的条件对这些组进行筛选，最终只输出满足指定条件的组，可以使用 HAVING 关键字指定筛选条件。

例如，查询收听用户总数大于 50 的用户 ID 信息：

```
SELECT user_id, COUNT(user_id) AS  收听用户数
FROM Z_USER_FUN
GROUP BY user_id
HAVING (COUNT(user_id) > 50)
```

技巧：

WHERE 子句与 HAVING 关键字的区别在于作用对象不同。WHERE 子句作用于表或视图，从中选择满足条件的元组。HAVING 关键字则作用于组，从中选择满足条件的组。

8. 连接查询

连接查询也叫多表查询，在实际应用过程中经常要同时从两个表或者两个以上的表中

检索数据。连接查询允许通过指定表中某个或者某些列作为连接条件，同时从两个表或者多个表中检索数据。

连接查询可以使用两种连接语法形式，一种是把连接条件写在 FROM 子句中，另一种是把连接条件写在 WHERE 子句中。

连接条件写在 WHERE 子句中的语法形式非常简单，在数据库程序中经常用到，语法格式如下：

```
SELECT 表名.字段名,表名.字段名,...
FROM 表名, 表名…
WHERE 连接条件 AND 搜索条件
```

连接条件一般是表与表之间联系字段的表达式，例如，下面的例子将查询“葛萌萌”发表的所有信息：

```
SELECT user_name, T.*
FROM Z_USER, Z_MESSAGE  T
WHERE  Z_USER.user_id = T.user_id AND  Z_USER.user_name = '葛萌萌'
```

技巧：

上述查询中，数据表 Z_MESSAGE 使用了别名 T，数据表别名功能能够大大减少手工输入量，使代码简洁明了。

连接条件写在 FROM 子句中的形式中需要用到 JOIN 关键字。

SQL-92 标准所定义的 FROM 子句的连接语法格式如下：

```
FROM join_table join_type join_table
[ON (join_condition)]
```

其中，join_table 指出参与连接操作的表名，连接可以对同一个表操作，也可以对多个表操作，对同一个表操作的连接又称做自连接；join_type 指出连接类型，可分为 3 种：内连接、外连接和交叉连接。

(1) 内连接(INNER JOIN)使用比较运算符进行表间某(些)列数据的比较操作，并列出这些表中与连接条件相匹配的数据行。根据所使用的比较方式不同，内连接又分为等值连接、自然连接和不等连接 3 种。

(2) 外连接分为左外连接(LEFT OUTER JOIN 或 LEFT JOIN)、右外连接(RIGHT OUTER JOIN 或 RIGHT JOIN)和全外连接(FULL OUTER JOIN 或 FULL JOIN)3 种。与内连接不同的是，外连接不只列出与连接条件相匹配的行，而是列出左表(左外连接时)、右表(右外连接时)或两个表(全外连接时)中所有符合搜索条件的数据行。

(3) 交叉连接(CROSS JOIN)没有 WHERE 子句，它返回连接表中所有数据行的笛卡尔积，其结果集合中的数据行数等于第一个表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。

连接操作中的 ON (join_condition) 子句指出连接条件，它由被连接表中的列和比较运算符以及逻辑运算符等构成。

注意：

无论哪种连接都不能对 text、ntext 和 image 数据类型进行直接连接，但可以对这三种列进行间接连接。

例如，上述查询“葛萌萌”发表的所有信息的 SQL 语句，可以转换为如下内连接：

```
SELECT Z_USER.user_name, Z_MESSAGE.*
FROM Z_USER INNER JOIN
        Z_MESSAGE ON Z_USER.user_id = Z_MESSAGE.user_id
WHERE (Z_USER.user_name = '葛萌萌')
```

9. 联合查询

UNION 运算符可以将两个或两个以上的 SELECT 语句的查询结果集合合并成一个结果集合显示，即执行联合查询。UNION 的语法格式如下：

```
select_statement
UNION [ALL] selectstatement
[UNION [ALL] selectstatement][…n]
```

其中，selectstatement 为待联合的 SELECT 查询语句；ALL 选项表示将所有行合并到结果集合中。不指定该项时，被联合查询结果集合中的重复行将只保留一行。

联合查询时，查询结果的列标题为第一个查询语句的列标题。因此，要定义列标题必须在第一个查询语句中定义。要对联合查询结果排序时，也必须使用第一查询语句中的列名、列标题或者列序号。

例如，下面的语句将返回姓名为“葛萌萌”和“赵艳铎”的用户信息：

```
SELECT user_id, user_name, user_sex
FROM Z_USER
WHERE (user_name = '葛萌萌')
UNION
SELECT user_id, user_name, user_sex
FROM Z_USER AS Z_USER_1
WHERE (user_name = '赵艳铎')
```

10. 子查询

在 SQL 中，一个 SELECT…FROM…WHERE 语句称为一个查询块。将一个查询块嵌套在另一个查询块的 WHERE 子句或 HAVING 关键字的条件中的查询称为嵌套查询。

SQL 语言允许多层嵌套查询，即一个子查询中还可以嵌套其他子查询。需要注意的是，子查询的 SELECT 语句中不能使用 ORDER BY 子句，因为 ORDER BY 子句只能对最终查询结果进行排序。

嵌套查询使得用多个简单查询构成复杂的查询成为可能，从而增强了 SQL 的查询能力。以层层嵌套的方式来构造程序正是 SQL 中“结构化”的含义所在。

- 带有 IN 谓词的子查询

在嵌套查询中，子查询的结果往往是一个集合，所以，IN 是嵌套查询中最经常使用的谓词。以下 SQL 语句查询“小石头”收听的所有用户的信息：

```
SELECT user_name, user_sex, user_email, user_address
FROM Z_USER
WHERE (user_id IN
          (SELECT Z_USER_FUN.fun_user_id
         FROM Z_USER AS Z_USER_1 INNER JOIN
               Z_USER_FUN ON Z_USER_1.user_id = Z_USER_FUN.user_id
         WHERE (Z_USER_1.user_name = '小石头')))
```

- 带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。当用户能确切知道内层查询返回的是单值时，可以用>、<、=、>=、<=、!=或<>等比较运算符。

以下 SQL 语句将查询“葛萌萌”发表的所有信息：

```
SELECT msg_id, user_id, msg_content, reply_count, post_time
FROM Z_MESSAGE
WHERE (user_id =
          (SELECT user_id
         FROM Z_USER
         WHERE (user_name = '葛萌萌')))
```

- 带有 ANY(SOME)或 ALL 谓词的子查询

子查询返回单值时可以用比较运算符，但返回多值时要用 ANY(有的系统用 SOME)或 ALL 谓词修饰符。而使用 ANY 或 ALL 谓词时则必须同时使用比较运算符，其语义定义如表 5-8 所示。

表 5-8　ANY 和比较运算符组合

运　算　符	意　　义
>ANY	大于子查询结果中的某个值
>ALL	大于子查询结果中的所有值
<ANY	小于子查询结果中的某个值
<ALL	小于子查询结果中的所有值
>=ANY	大于等于子查询结果中的某个值
>=ALL	大于等于子查询结果中的所有值
<=ANY	小于等于子查询结果中的某个值
<=ALL	小于等于子查询结果中的所有值
=ANY	等于子查询结果中的某个值
=ALL	等于子查询结果中的所有值
!=(或<>)ANY	不等于子查询结果中的某个值
!=(或<>)ALL	不等于子查询结果中的任何一个值

以下 SQL 语句查询所有收听用户数小于最近 24 小时注册的所有新用户收听用户数的用户的收听情况：

```
SELECT user_id, COUNT(user_id) AS 收听用户数
FROM Z_USER_FUN
GROUP BY user_id
HAVING (COUNT(user_id) < ALL
          (SELECT COUNT(user_id) AS Expr1
         FROM Z_USER_FUN AS Z_USER_FUN_1
         WHERE (user_id IN
                   (SELECT user_id
                  FROM Z_USER
                  WHERE (regist_time > GETDATE() - 1)))
         GROUP BY user_id))
```

系统在执行此查询语句时，首先处理子查询，找出最近 24 小时新注册的所有用户，然后计算出这些用户各的自收听用户数，构成一个集合，最后，通过 HAVING 子句和<ALL 关键字筛选出比这些新用户收听还少的用户的收听用户数。

5.2.3　在 VWD 中执行 SQL 查询

5.2.2 节介绍了 SELECT 语句的语法，现在来看看，如何在 VWD 中执行这些查询语句。

例 5-5：在 VWD 中，通过 SELECT 语句查询表中的数据。

(1) 前面使用了“显示表数据”命令，查看表中的数据，并且可以向表中添加数据。这实际上是执行的一个 SELECT 查询的结果。要查看出现这一列表所涉及的查询，可以使用“查询设计器”工具栏中的命令按钮。如果该工具栏不可见，可以右击已有工具栏，并从打开的快捷菜单中选择“查询设计器”选项即可打开，工具栏中个命令按钮的作用如图 5-17 所示。

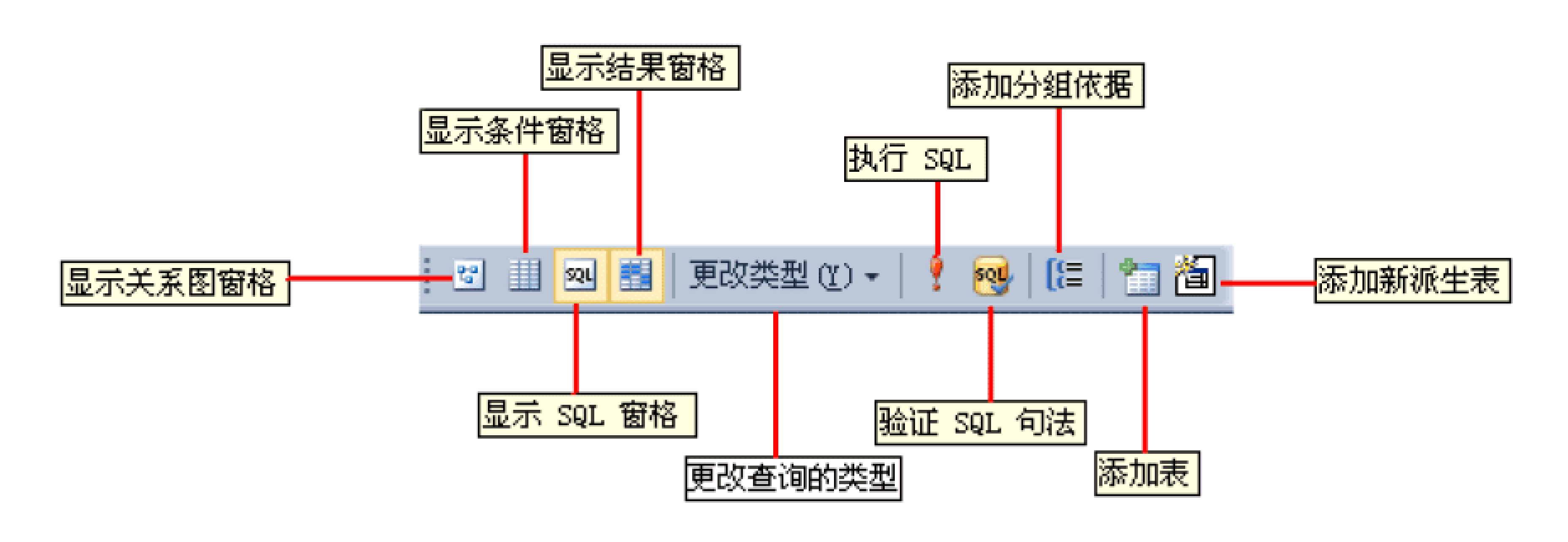


图 5-17　“查询设计器”工具栏

(2) 用户可以看到表数据窗口，其实是显示的结果窗格，在该工具栏中，单击“显示关系图窗格”、“显示条件窗格”和“显示 SQL 窗格”按钮分别打开这些窗口。则文档窗口将被分为 4 个区域，每个区域对应于一个窗格，如图 5-18 所示。

其中的 SQL 窗格显示了用于检索显示在结果窗格中的 SQL 语句。在这里可以很轻松地修改它，以执行不同的查询。

(3) 为了确保 SQL 语句是有效的，可单击工具栏中的“验证 SQL 句法”按钮，修正 SQL 语句可能包含的错误。然后单击“执行 SQL”按钮或按 Ctrl+R 组合键。在这两种情况下，都会执行 SQL 语句，并更新结果窗格。

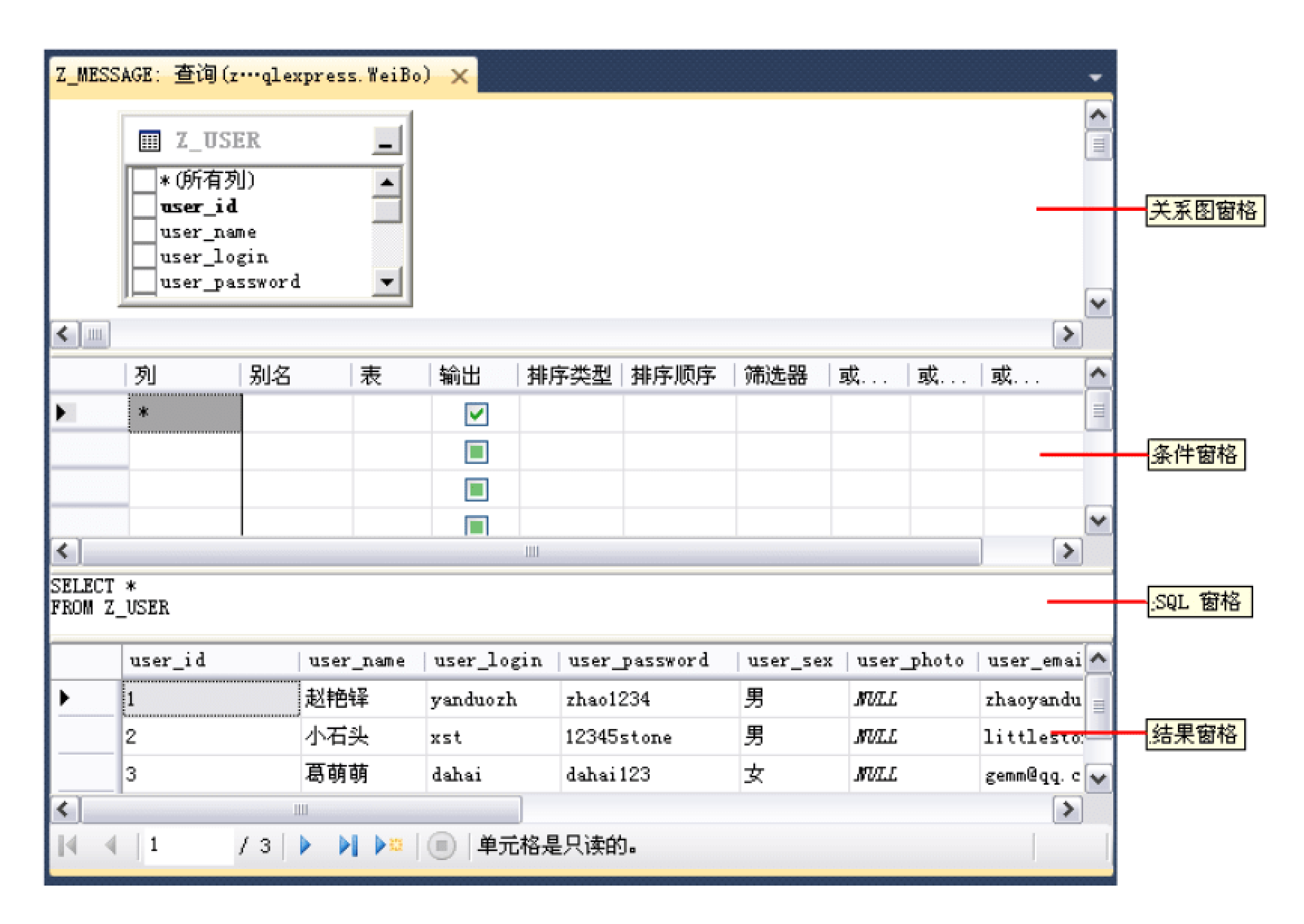


图 5-18　显示所有窗格

(4) 除了直接在“SQL 窗格”中输入 SQL 语句之外，还可以通过“关系图窗格”和“条件窗格”来控制输出结果中包含的字段以及查询的过滤条件和排序方式等。而且在这两个窗格中的修改会马上体现在 SQL 窗格的 SQL 语句上。例如，改变输出列，然后设置排序条件和筛选条件后，可以看到 SQL 语句也相应地变化了，如图 5-19 所示。

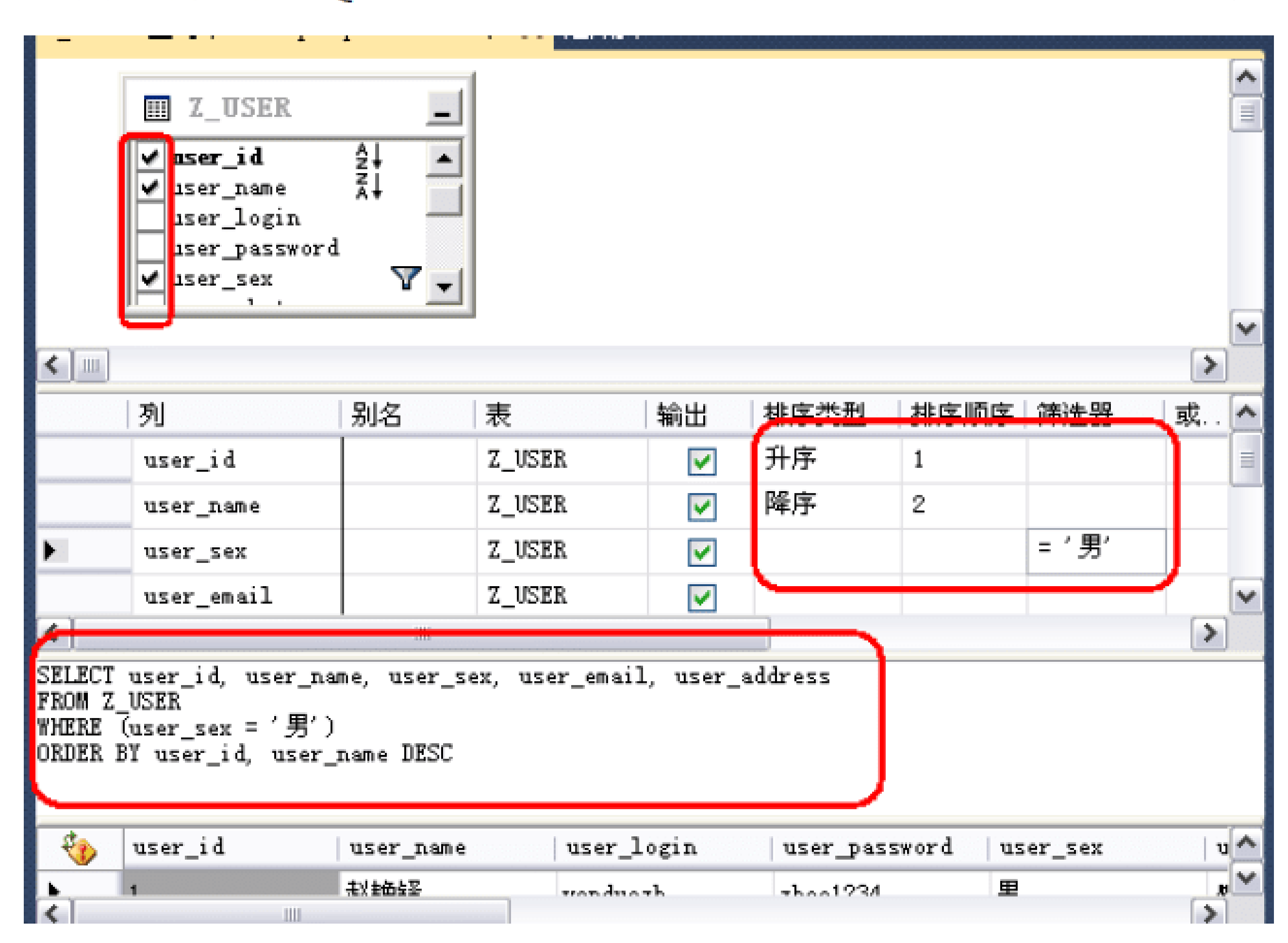


图 5-19　改变条件窗格自动生成相应的 SQL

(5) 再次按 Ctrl+R 组合键或单击工具栏中的“执行 SQL”按钮，结果窗格将显示新的查询记录。

(6) 如果要进行多表查询，可以单击工具栏中的“添加表”按钮添加所需的表。此时，SQL 窗格中会根据表之间的关系自动生成连接查询语句，读者可根据实际需要进行相应的修改。

5.2.4　INSERT 语句

要将新记录插入到表中，可以使用 INSERT 语句。它有一些不同的形式，最简单的形式如下所示：

```
INSERT INTO  表名
VALUES (第一个字段值,...,最后一个字段值)
```

其中，VALUES 后面的字段值必须与数据表中相应字段所规定的值得数据类型相符，和 WHERE 子句一样，需要将字符串和日期值括在单引号中，但可以在 SQL 语句中直接输入数字和布尔值。如果不想对某些字段进行赋值，可以在相应地位置上用空值 NULL 替代。

如果需要插入的是表的某些字段的值，可以在 SQL 语句中使用另一种 INSERT 语句进行操作，语法格式如下：

```
INSERT INTO  表名(字段 1,…,字段 N,…)
VALUES (第一个字段值,...,第 N 个字段值,…)
```

当用这种形式向数据表中添加新记录时，在关键字 INSERT INTO 后面输入所要添加的数据表名称，然后在括号中列出将要添加新值的字段的名称，最后，在关键字 VALUES 的后面按照前面输入的列的顺序对应地输入所有要添加的记录值。需要注意的是，对于表中的非空列，如果没有定义默认值，必须显示的给出该列的值。

例如，下面的语句将在 Z_MESSAGE 表中插入一条新的记录：

```
INSERT INTO Z_MESSAGE
        ( user_id, msg_content, reply_count,post_time)
VALUES (1, '既然已移情，那么请别恋', 0, '2012-7-7 22:22:22')
```

技巧：

在 VWD 的查询窗口中，单击工具栏中的“更改类型”按钮，从弹出的下拉菜单中选择“插入值”选项，SQL 查询中的语句会自动更新为 INSERT 语句的模板，可通过“条件窗格”指定各列的值，或者直接基于自动生成的 INSERT 模板完成插入语句。

5.2.5　UPDATE 语句

UPDATE 语句用来修改数据表中已经存在的数据记录。它的基本语法格式如下：

```
UPDATE   表名
SET  字段 1 = 值 1,…, 字段 N = 值 N,
［WHERE 条件表达式］
```

其含义是更新数据表中符合 WHERE 条件的字段或字段集合的值，WHERE 条件是可选的。例如，下面的语句是下是将用户名为“葛萌萌”的 user_email 更改为 lovezyd@gmail.com：

```
UPDATE   Z_USER
SET user_email =   'lovezyd@gmail.com'
WHERE user_name = '葛萌萌'
```

上面的 UPDATE 语句是确定值赋值，即为字段赋予指定的值。也可以基于已有的值来设置新的字段值。例如，将信息表 Z_MESSAGE 中 msg_id 为 1 的记录的 reply_count 字段加 1，可以使用如下 SQL 语句：

```
UPDATE Z_MESSAGE
SET reply_count = reply_count + 1
WHERE msg_id = 1
```

与插入数据一样，单击工具栏中的“更改类型”按钮，从弹出的下拉菜单中选择“更新”选项，SQL 查询中的语句会自动更新为 UPDATE 语句的模板，可通过“条件窗格”或者直接在 SQL 窗格中输入语句来完成更新操作。

5.2.6　DELETE 语句

DELETE 语句用来删除数据表中的记录，基本语法格式如下：

```
DELETE   FROM 表名
[WHERE 条件表达式]
```

与 UPDATE 语句类似，DELETE 语句中的 WHERE 选项也是可选的，如果不限定 WHERE 条件，DELETE 语句将删除数据表中的所有记录，例如：

```
DELETE FROM Z_USER
```

这条语句将删除用户表 Z_USER 中的所有记录。如果并不想清空数据表中所有记录，就必须注意 WHERE 子句的使用。

技巧：

单击工具栏中的“更改类型”按钮，从弹出的下拉菜单中选择“删除”选项，SQL 查询中的语句会自动更新为 DELETE 语句的模板。

5.3　使用 ADO.NET

ADO.NET 是.NET Framework 提供的数据访问的类库，ADO.NET 对 Microsoft SQL Server、Oracle 和 XML 等数据源提供一致的访问。应用程序可以使用 ADO.NET 连接到这

些数据源，并检索和更新所包含的数据。

5.3.1　ADO.NET 概述

ADO.NET 作为重要的.NET 数据库应用程序的解决方案，更多地显示了涵盖全面的设计，而不仅是单纯地以数据库为中心，如图 5-20 所示为 ADO.NET 的架构。

在 ADO.NET 中，访问 ADO.NET 中的数据源是由托管提供程序所控制的，ADO.NET 架构的两个主要组件是 Data Provider(数据提供者)和 DataSet(数据集)。相应地可以把 ADO.NET 的基本类分为数据提供者对象和数据集对象。提供者对象可用于每一种类型的数据源；专用于提供者的对象完成数据源中实际的读取和写入工作。而数据集对象则是将数据读入到内存中，用来访问和操纵数据。如图 5-21 所示为这些对象之间的关系。

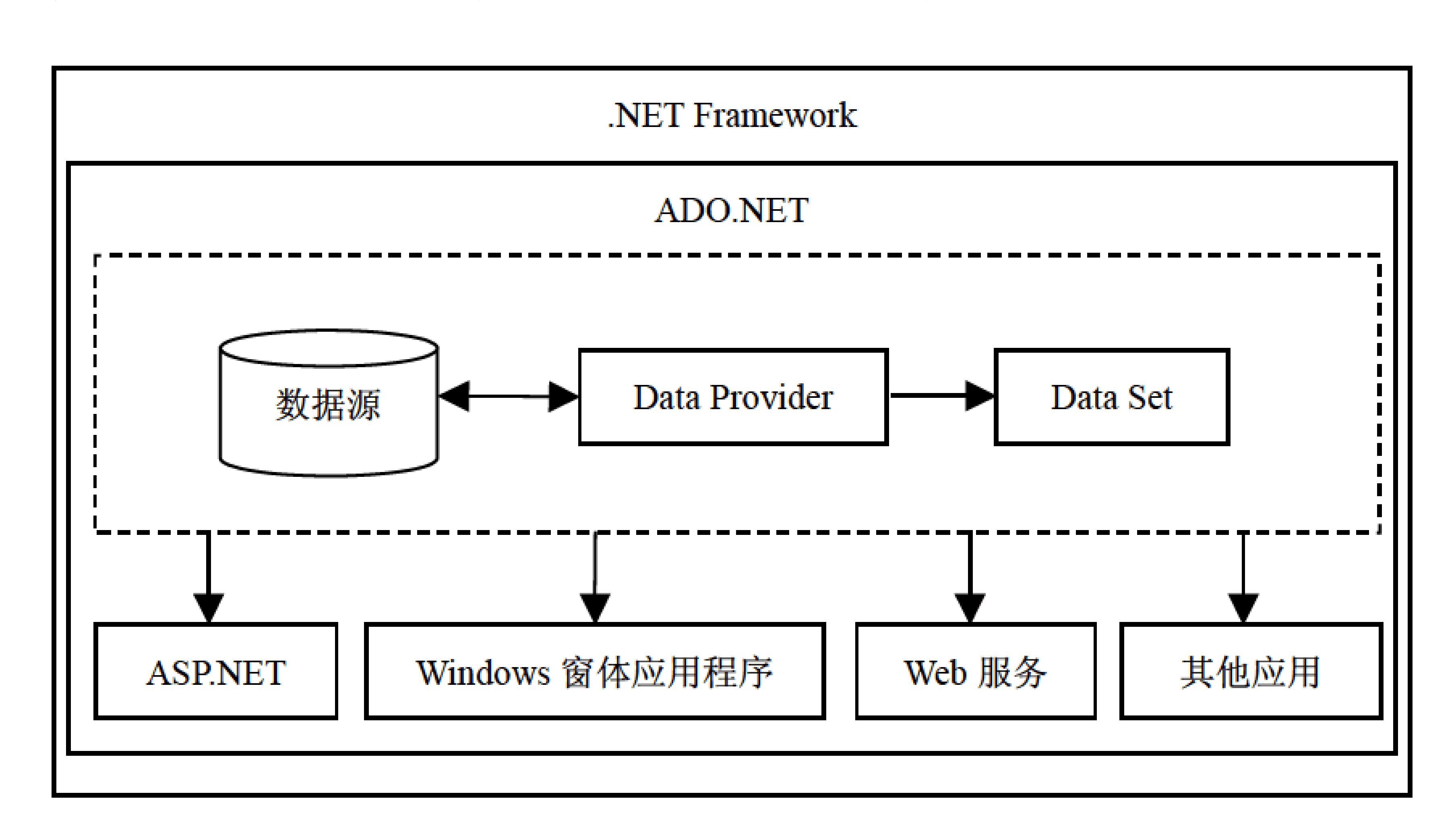


图 5-20　ADO.NET 架构

提供者对象需要一个活动的连接，可以使用它们先从数据库中读取数据，然后根据需要，通过数据集对象使用内存中的数据，也可以使用提供者对象更新数据源中的数据。数据集对象以非连接方式使用；甚至在数据库连接关闭之后，也可以使用内存中的数据。

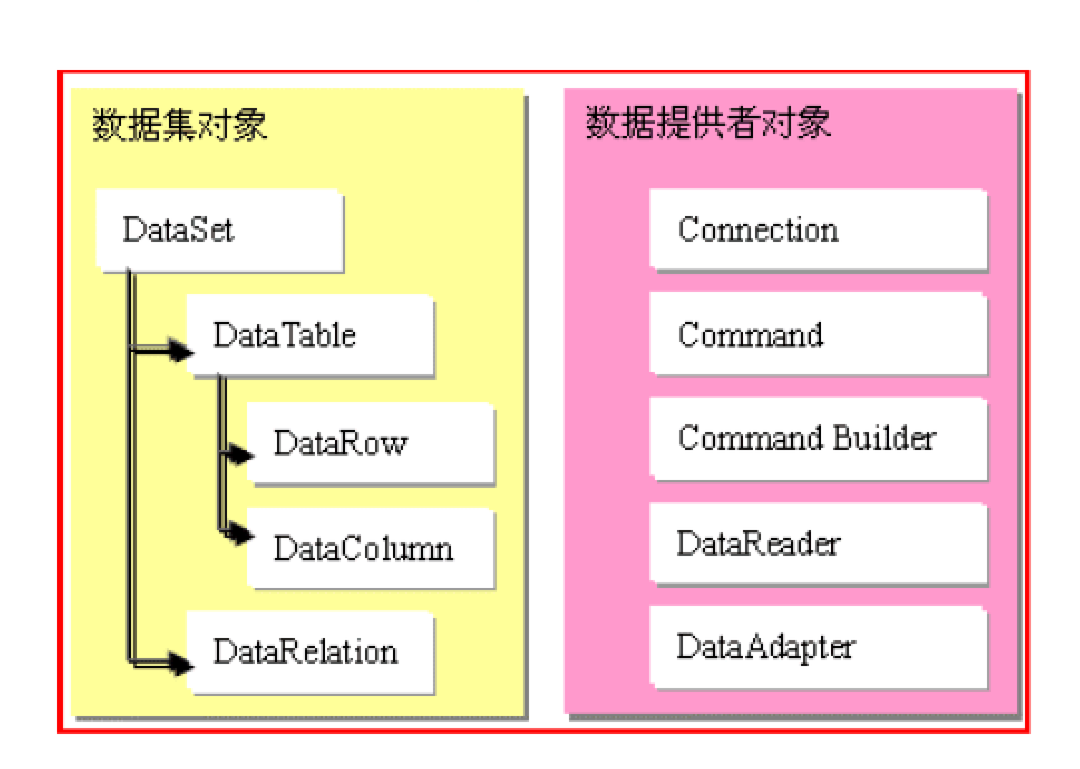


图 5-21　ADO.NET 类之间的关系

5.3.2　提供者对象

Data Provider 提供了 DataSet 和数据中心(如 MS SQL Server)之间的联系，同时也包含了存取数据中心(数据库)的一系列接口。通过数据提供者所提供的应用程序编程接口(API)，可以轻松访问各种数据源的数据，包括 OLE DB 所支持的和 ODBC 支持的数据库。

提供者对象就是指在每一个.NET 数据提供者中定义的对象，其名称前带有特定提供者的名称。因此，用于 OLE DB 提供者的连接对象就是 OleDbConnection；用于 SQL Server .NET 提供者的类就是 SqlConnection。

在 ADO.NET 中，连接数据源有 4 种接口，即 SQLClient、OracleClient 、ODBC 和 OLEDB。其中，SQLClient 是 Microsoft SQL Server 数据库专用连接接口，OracleClient 是 Oracle 数据库专用的连接接口，ODBC 和 OLEDB 可用于其他数据源的连接。在应用程序中使用任何一种连接接口时，必须在后台代码中引用对应的命名空间，而且类的名称也随之发生变化，如表 5-9 所示。

表 5-9　数据连接方式命名空间与对应的类名称

名 称 空 间	对应的类名称
System.Data.SqlClient	SqlConnection、SqlCommand、SqlDataReader、SqlDataAdapter
System.Data.Odbc	OdbcConnection、OdbcCommand、OdbcDataReader、OdbcDataAdapter
System.Data.OleDb	OleDbConnection、OleDbCommand、OleDbDataReader、OleDbDataAdapter
System.Data.OracleClientt	OracleConnection、OracleCommand、OracleDataReader、OracleDataAdapter

1. 连接对象

连接对象是使用 ADO.NET 访问数据库的第一个对象，它提供了到数据源的基本连接。如果使用的数据库需要用户名和密码，或者是位于远程网络服务器上，则连接对象可以提供建立连接并登录的细节。根据数据源的不同，连接对象有 4 种，即 SqlConnection、OleDbConnection、OdbcConnection 和 OracleConnection。

连接对象的常用属性和方法如表 5-10 所示。

表 5-10　连接对象的常用属性和方法

属性或方法	描　　述
ConnectionString	该属性用来指定连接的数据源，需要使用很多参数：如 Data Source 指明数据源；Initial Catalog 指明数据库；Integrated Security 指明集成安全；User ID 和 Password 分别用于指明登录账户和密码等
ConnectionTimeout	该属性用于获取在尝试建立连接时终止尝试并生成错误之前所等待的时间，单位为秒，默认值为 15

(续表)

属性或方法	描　述
Database	该属性返回当前数据库名称或连接打开后要使用的数据库名称，默认为空字符串
DataSource	获取要连接的数据源实例的名称
Open	该方法用于打开由 ConnectionString 属性指定的数据源连接
Close	该方法用于断开由 ConnectionString 属性指定的数据源连接

2. 命令对象

命令对象用于向数据源发出命令，命令对象可直接执行 SQL 语句或存储过程，命令对象的 CommandText 属性就是要执行的 SQL 语句，如 SELECT * FROM Z_REPLY。对于不同的提供者，该对象的名称也略有不同，例如，用于 SQL Server 的命令对象为 SqlCommand，用于 ODBC 的为 OdbcCommand，用于 OLE DB 的命令对象为 OleDbCommand，用于 Oracle 的命令对象为 OracleCommand。

Command 对象的常用属性和方法如表 5-11 所示。

表 5-11　Command 对象的常用属性和方法

属性或方法	描　述
CommandText	获取或设置要对数据源执行的 SQL 语句或存储过程
CommandTimeout	获取或设置在终止执行命令的尝试并生成错误之前的等待时间，以秒为单位，默认值为 30
CommandType	获取或设置 CommandText 的类型，其值为 System.Data.CommandType 值之一，默认为 Text
Connection	获取或设置 Command 实例使用的 Connection 对象
ExcuteNonQuery	该方法对连接的数据库执行 SQL 语句并返回受影响的行数
ExcuteReader	将 CommandText 发送到 Connection 并生成一个 DataReader，返回值为一个 DataReader 对象
ExcuteScalar	执行查询并返回查询所返回的结果集中的第一行第一列，忽略其他行或列
ExcuteXmlReader	将 CommandText 发送到 Connection 并生成一个 System.Xml.XmlReader 对象，返回值为该 XmlReader 对象
ResetCommandTimeout	将 CommandTimeout 属性重置为默认值

3. CommandBuilder 对象

此对象用于构建 SQL 命令，在基于单一表查询的对象中进行数据修改。对于不同的提供者，该对象的名称分别为用于 SQL Server 的 SqlCommandBuilder，用于 ODBC 的 Odbc-

CommandBuilder，用于 OLE DB 的 OleDbCommandBuilder 和用于 Oracle 的 OracleCommand-Builder。

CommandBuilder 对象的常用属性和方法如表 5-12 所示。

表 5-12　CommandBuilder 对象的常用属性和方法

属性或方法	描　　述
DataAdapter	获取或设置自动为其生成 SQL 语句的一个 DataAdapter 对象
GetUpdateCommand	获取自动生成的、对数据库执行更新操作所需的 Command 对象
GetDeleteCommand	获取自动生成的、对数据库执行删除操作所需的 Command 对象
GetInsertComman	获取自动生成的、对数据库执行插入操作所需的 Command 对象

4. DataReader 对象

该对象用于从数据源中读取仅能前向和只读的数据流。对于简单地读取数据来说，此对象的性能最好。对于不同的提供者，该对象的名称分别为用于 SQL Server 的 SqlDataReader，用于 ODBC 的 OdbcDataReader，用于 OLE DB 的 OleDbDataReader 和用于 Oracle 的 OracleDataReader。

DataReader 对象的常用属性和方法如表 5-13 所示。

表 5-13　DataReader 对象的常用属性和方法

属性或方法	描　　述
FieldCount	获取当前行中的列数
RecordsAffected	获取被更改、插入或删除的行数
IsClosed	指示是否可关闭数据读取器
Close	关闭 DataReader 对象
GetName	获取指定列的名称
Read	使 DataReader 前进到下一条记录，如果存在多个行则返回 true，否则返回 false
NextResult	当读取批处理 SQL 语句的结果时，使数据读取器前进到下一个结果，如果存在多个结果集则返回 true，否则返回 false
IsDBNull	获取一个值，指示列中是否包含不存在的或已丢失的值
GetOrdinal	在给定列名称的情况下获取列序号

5. DataAdapter 对象

DataAdapter(数据适配器)是 DataSet 和数据源之间的桥梁，可以执行针对数据源的各种操作，包括更新变动的数据、填充 DataSet 对象以及其他操作。对于不同的提供者，该对象的名称分别为用于 SQL Server 的 SqlDataAdapter，用于 ODBC 的 OdbcDataAdapter、用于 OLE DB 的 OleDbAdapter 和用于 Oracle 的 OracleDataAdapter。

在创建 DataAdapter 对象时，可以直接指定 Connection 和 Command 对象。如果要定义后指定属性，主要包括 SelectCommand、InsertCommand、UpdateCommand 和 DeleteCommand。

DataAdapter 对象的常用方法主要有如下 3 个。

- Fill：该方法用来执行 SelectCommand，用数据源的数据填充 DataSet 对象。
- GetData：该方法新建一个数据集中 DataTable 并填充它。
- Update：更新数据集中的某个 DataTable。

5.3.3　数据集对象

DataSet 即数据集，是 ADO.NET 的断开式结构的核心，是指内存中的数据库数据的副本，用于支持 ADO.NET 中的离线数据的访问。DataSet 对象表示了数据库中的完整的数据，包括表、限制以及表之间的关系。正是由于 DataSet 的存在，才使得编程人员在编写应用程序时可以不考虑各数据源之间的差异，从而使用统一的编程接口。

运行时，组件可以交换数据集。也就是说，一个组件可以将数据集传递给另一个组件。为了适应在组件间进行数据集交换，ADO.NET 使用了一个基于 XML 的保持和传递格式。ADO.NET 解决方案将内存中的数据(数据库)表示为一个 XML 文件，然后将这个 XML 文件发送给另一个组件。

数据集对象位于 System.Data 命名空间中，用于定义 ADO.NET 的断开的、客户端的对象，包括 DataSet、DataTable、DataRow、DataColumn 和 DataRelation 等。

用户可以使用 DataSet 对象对数据集中的内容进行处理。DataSet 对象允许使用与关系模型一致的方法对数据集的内容进行处理。例如，DataSet 对象有一个 DataTable 对象集合，每个 DataTable 对象都有行和列，并且与其他的 DataTable 对象有关联。当一个组件将数据集传递给另一个组件时，接收组件将把接收到的数据集物化为一个 DataSet 对象。

1. DataSet 对象

DataSet 是数据集对象中的首要对象，此对象表示一组相关表，在应用程序中这些表作为一个单元来引用。例如，Z_USER、Z_MESSAGE 和 Z_REPLY 是一个 DataSet 中的 3 张表，有了此对象，就可以快速从每一个表中获取所需要的数据，当与服务器断开时检查并修改数据，然后在另一个操作中使用这些修改的数据更新服务器。

DataSet 允许访问低级对象，这些对象代表单独的表和关系。这些对象是 DataTable 对象和 DataRelation 对象。

DataSet 对象的常用方法如表 5-14 所示。

表 5-14　DataSet 对象的常用方法

属性或方法	描　　述
AcceptChanges	提交自加载此 DataSet 或上次调用 AcceptChanges 以来对其进行的所有更改
BeginInit	开始初始化在窗体上使用或由另一个组件使用的 DataSet，初始化发生在运行时

(续表)

属性或方法	描　　述
EndInit	结束在窗体上使用或由另一个组件使用的 DataSet 的初始化
Clear	通过移除所有表中的所有行来清除所有数据的 DataSet
Clone	复制 DataSet 的结构，包括所有 DataTable 架构、关系和约束。不复制任何数据
Copy	复制 DataSet 的结构和数据，返回新的 DataSet,具有与该 DataSet 相同的结构和数据
Merge	将指定的 DataSet 及其架构合并到当前 DataSet 中
GetChanges	获取 DataSet 的副本，该副本包含自加载以来或上次调用 AcceptChanges 方法以来所有的更改，可以对该副本执行操作

2. DataTable 对象

该对象代表 DataSet 中的一个表，如 Z_USER。

DataTable 对象的 Rows 和 Columns 分别是 DataRow 和 DataColumn 对象，可用于访问 DataTable 表中的行和列。

- DataColumn 对象：代表表中的一列，如 user_name 或 user_login。
- DataRow 对象：代表来自表的关联数据的一行，如 Z_USER 表中的 user_id、user_name 和 user_address 等。

3. DataRelation 对象

该对象代表通过共享列而发生关系的两个表之间的关系，如 Z_MESSAGE 表中的 user_id 列标识发表信息的用户。于是，可以创建 DataRelation 对象，通过共享列 user_id 建立 Z_USER 和 Z_MESSAGE 表之间的关系。

5.3.4　使用 ADO.NET 访问数据库

ASP.NET 数据访问程序的开发流程有以下几个步骤。

(1) 利用 Connection 对象创建数据连接。

(2) 利用 Command 对象数据源执行 SQL 命令。

(3) 利用 DataReader 对象读取数据源的数据。

(4) DataSet 对象与 DataAdapter 对象配合，完成数据的查询和更新操作。

为了让读者更深刻的理解 ADO.NET 访问数据库的用法，下面将以具体的实例来介绍对数据的查询、插入、修改和删除操作。

1. 注册新用户

例 5-6：新建一个 ASP.NET 页面，完成注册微博新用户功能。

注册新用户就是向 Z_USER 表中添加一条新记录，在添加记录之前需要判断是否已经存在相同的用户登录名，并且要求用户设置的个人微博地址 home_url 唯一。

(1) 启动 VWD 2010，选择“文件”|“新建网站”命令，新建空网站 Chapter5。

(2) 通过“添加新项”对话框添加一个名为 Register.aspx 的页面。

(3) 使用表格布局，插入一个 11 行 3 列的表格，合并表格第 1 行的 3 个单元格，后面的第 1 列输入文本信息，第 2 列添加相应的控件，第 3 列添加验证控件，最后一行添加 2 个按钮控件，用来提交和取消输入。最终的页面布局如图 5-22 所示。

图 5-22　注册页面的布局

这里的注册页面只选取了 Z_USER 表中的非空项和部分可空字段要求用户输入，其他信息可以在用户注册并成功登录后，通过完善个人资料和上传头像等功能页面实现。

(4) 打开页面的后台代码文件 Register.aspx.cs，由于要访问数据库，所以在后台代码文件中需要引入相应的命名空间：

```
using System.Data.SqlClient;
using System.Data;
```

(5) 创建连接对象时，需要设置连接字符串属性 ConnectionString，通常将该连接串配置在 web.config 配置文件中，打开该文件，添加如下配置信息：

```
    <connectionStrings>
        <add name="WeiBoConnectionString" connectionString="Data
Source=zhao\sqlexpress;Initial Catalog=WeiBo;Integrated Security=True"
            providerName="System.Data.SqlClient" />
    </connectionStrings>
```

(6) 在第 2 章已经介绍过读取 web.config 文件中的配置项，因此要先添加 using 语句引入 System.Web.Configuration;命名空间。

(7) 接下来添加一个私有方法，验证登录名和个人微博地址是否已经被使用，该方法有两个参数，第 1 个参数是要验证的字段值，第 2 个参数是验证类别，该参数有个默认值 LOGIN，如果不指定该参数则进行登录名验证，如果该参数为 URL，则进行个人微博地址验证。代码如下：

```
        private bool doCheck(string condValue,string type="LOGIN")
        {
            bool ret = true;
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Z_USER WHERE user_login
= @cond_value", con);
            if(type.Equals("URL"))
                cmd.CommandText = "SELECT * FROM Z_USER WHERE home_url =
@cond_value";
            cmd.Parameters.AddWithValue("@cond_value", condValue);
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())//如果有结果表明记录已存在
            {
                ret= false;
            }
            cmd = null;
            con.Close();
            con = null;
            return ret;
        }
```

(8) 为“提交”按钮添加单击事件处理程序，代码如下：

```
        protected void ButtonSubmit_Click(object sender, EventArgs e)
        {
            if (!doCheck(TextBoxLogin.Text))
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"该登
录名已存在\"); ", true);
                return;
            }
            if (!doCheck(TextBoxHomeUrl.Text,"URL"))
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"个人
微博地址已被使用\"); ", true);
                return;
            }
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("INSERT INTO Z_USER(user_name," +
                "user_login,user_password,user_sex,home_url,user_email,user_info," +
```

```
                "user_address,regist_time) values(@user_name,@user_login,@user_password," +
                "@user_sex,@home_url,@user_email,@user_info,@user_address,@regist_time)",
con);
            cmd.Parameters.AddWithValue("@user_name", TextBoxName.Text);
            cmd.Parameters.AddWithValue("@user_login", TextBoxLogin.Text);
            cmd.Parameters.AddWithValue("@user_password", TextBoxPassword.Text);
            cmd.Parameters.AddWithValue("@user_sex", RadioButtonList1.SelectedValue);
            cmd.Parameters.AddWithValue("@home_url", TextBoxHomeUrl.Text);
            cmd.Parameters.AddWithValue("@user_email", TextBoxEmail.Text);
            cmd.Parameters.AddWithValue("@user_info", TextBoxInfo.Text);
            cmd.Parameters.AddWithValue("@user_address", TextBoxAddress.Text);
            cmd.Parameters.AddWithValue("@regist_time", DateTime.Now);
            int count = cmd.ExecuteNonQuery();
            if (count == 1)//如果为 1  表示插入 1 条记录
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"恭喜
你，注册成功\"); ", true);
            else
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"注册
失败\"); ", true);

            cmd = null;
            con.Close();
            con = null;
            Response.Redirect("Login.aspx");
        }
```

(9) 编译并运行程序，在浏览器中打开 Register.aspx。输入注册信息，如果必填项没有输入，验证控件会给出相应的提示信息。输入完整信息后，如图 5-23 所示，单击“提交”按钮，将验证登录名和个人微博地址是否可用，如果可用即可注册成功，并跳转到登录页面 Login.aspx。此时浏览器会提示找不到资源，因为还没有创建该页面，本章下一个例子就是实现微博的登录页面 Login.aspx。

提示：

注册成功后，读者可到数据库中查看 Z_USER 表，看是否新增了记录。

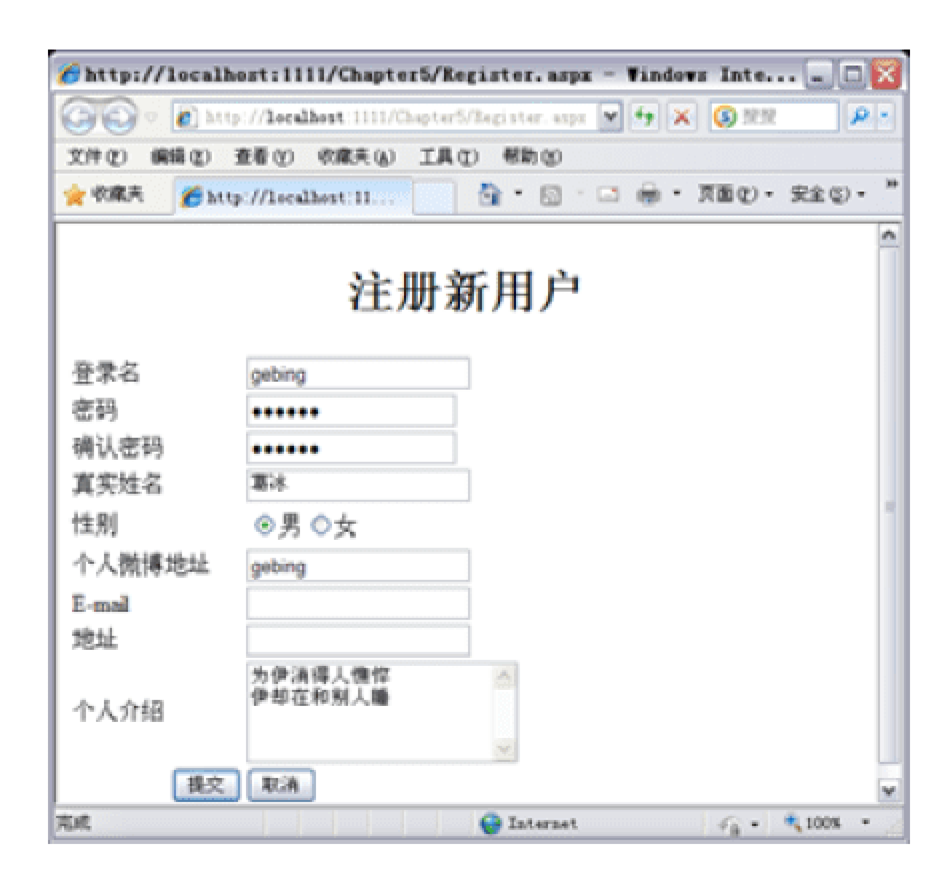


图 5-23　注册页面效果

2. 登录微博

例 5-7：新建一个 ASP.NET 页面，实现微博用户的登录功能，用户登录成功后显示用户的基本信息、用户收听和被收听的总数以及用户发表的信息数(只显示数量，不显示具体内容，待学习了数据控件后将显示信息列表)。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”对话框添加一个名为 Login.aspx 的页面。

(3) 在页面中添加一个 Panel 控件，该控件中用表格布局，添加相应的控件用于输入登录名和密码，并为两个文本框分别添加必选项验证控件，Panel1 控件的布局如图 5-24 所示。

(4) 添加另一个 Panel 控件 Panel2，用于登录成功后显示用户相关信息。控件中包括用户头像和基本信息，布局如图 5-25 所示。

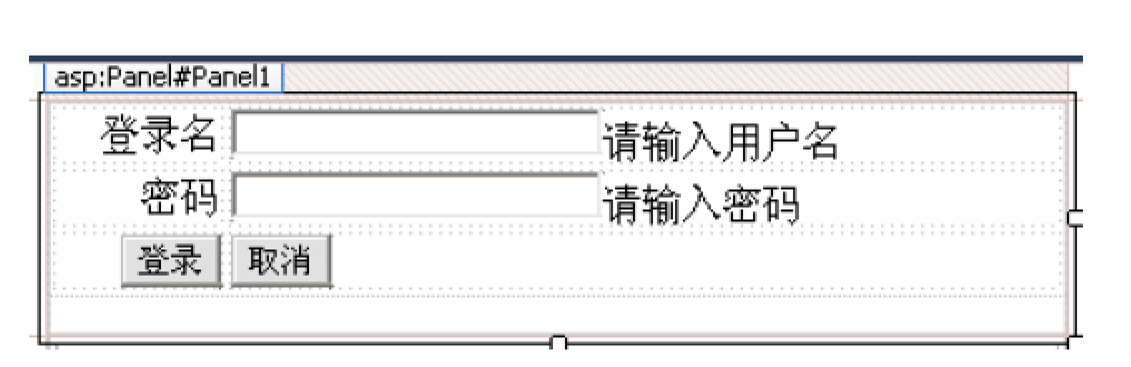


图 5-24　Panel1 控件布局

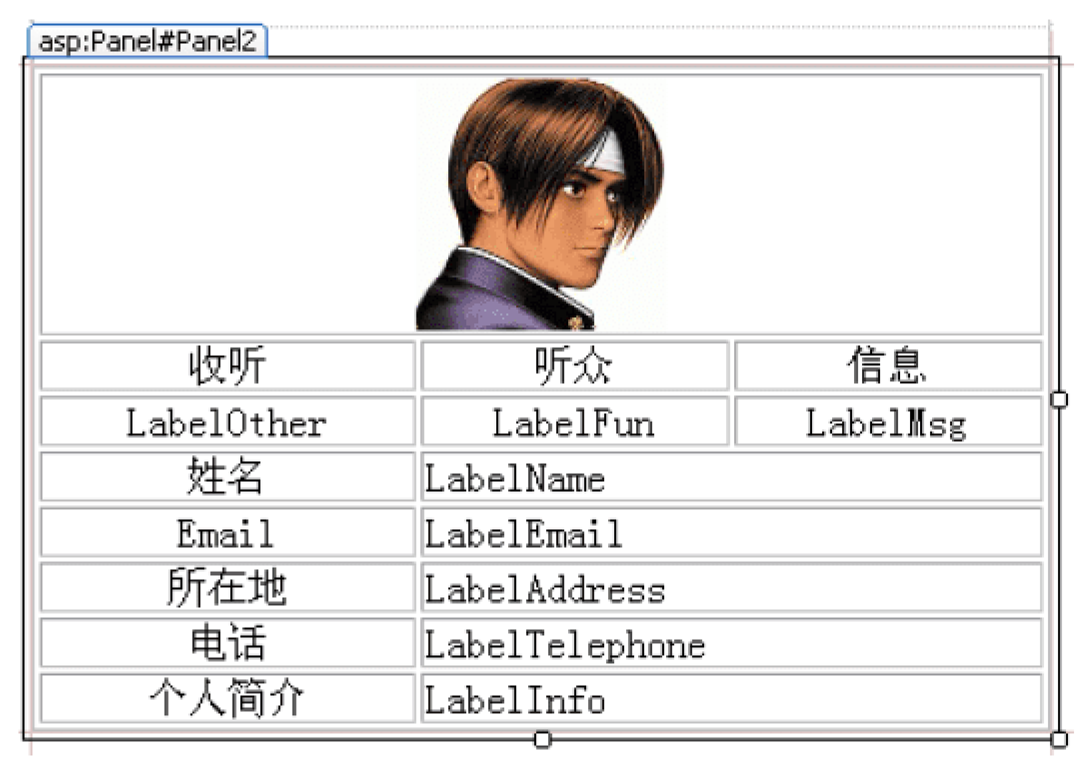


图 5-25　Panel2 控件布局

(5) 打开页面的后台代码文件 Login.aspx.cs，添加 using 语句，引入所需的命名空间：

```
using System.Web.Configuration;
using System.Data.SqlClient;
using System.Data;
```

(6) 为“登录”按钮添加单击事件处理程序，代码如下：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
        SqlConnection con = new SqlConnection(strConn);
        DataSet ds = new DataSet();
        con.Open();
        SqlDataAdapter sqld = new SqlDataAdapter("SELECT user_id,user_name FROM Z_USER
WHERE "+
            " user_login = @user_login AND user_password = @password", con);
        sqld.SelectCommand.Parameters.AddWithValue("@user_login", TextBoxLogin.Text);
        sqld.SelectCommand.Parameters.AddWithValue("@password", TextBoxPassword.Text);
        sqld.Fill(ds, "user");//用 Fill 方法填充 DataSet
        DataTable dTable = ds.Tables["user"];//将数据表的数据复制到 DataTable 对象
```

```
            DataRowCollection rows = dTable.Rows;//获取数据行
            if (rows.Count > 0)
            {
                Session["user_id"] = rows[0][0];
                Session["user_name"] = rows[0][1];
                ShowInfo(Int32.Parse(rows[0][0].ToString()));
            }
            else
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"用户
名密码错误\"); ", true);
            }
            con.Close();
            con = null;
        }
```

上述代码与注册页面所用的方法略有不同，这里使用的是 DataAdapter 对象和 DataSet 对象来读取数据集中的数据。感兴趣的读者也可以将其改为用 DataReader 对象实现。

(7) 登录成功后，将用户的 ID 和姓名存放到 Session 变量中，然后调用私有函数 ShowInfo 更新 Panel2 中的控件信息并显示，相应的代码如下所示：

```
        private void ShowInfo(int user_id)
        {
            Panel1.Visible = false;
            Panel2.Visible = true;
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT
user_name,user_email,user_address,user_info"+
                ",user_telephone FROM Z_USER WHERE user_id = "+user_id.ToString(), con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//如果有结果表明记录已存在
            {
                LabelName.Text=reader.GetString(0);
                if (!reader.IsDBNull(1))
                    LabelEmail.Text = reader.GetString(1);
                else
                    LabelEmail.Text = "";
                if (!reader.IsDBNull(2))
                    LabelAddress.Text = reader.GetString(2);
                else
                    LabelAddress.Text = "";
                if (!reader.IsDBNull(3))
```

```
                LabelInfo.Text = reader.GetString(3);
            else
                LabelInfo.Text = "";
            if (!reader.IsDBNull(4))
                LabelTelephone.Text = reader.GetString(4);
            else
                LabelTelephone.Text = "";
        }
        cmd.CommandText = "SELECT COUNT(*) FROM Z_USER_FUN WHERE user_id = " +
user_id.ToString();
        reader.Close();
        reader = cmd.ExecuteReader();
        if (reader.Read())
            LabelOther.Text = reader.GetInt32(0).ToString();
        cmd.CommandText = "SELECT COUNT(*) FROM Z_USER_FUN WHERE fun_user_id =
" + user_id.ToString();
        reader.Close();
        reader = cmd.ExecuteReader();
        if (reader.Read())
            LabelFun.Text = reader.GetInt32(0).ToString();
        cmd.CommandText = "SELECT COUNT(*) FROM Z_MESSAGE WHERE user_id = " +
user_id.ToString();
        reader.Close();
        reader = cmd.ExecuteReader();
        if (reader.Read())
            LabelMsg.Text = reader.GetInt32(0).ToString();
        reader.Close();
        cmd = null;
        con.Close();
        con = null;
        ShowPhoto(user_id);//显示头像
    }
```

该方法通过几次查询操作获取 Panel2 中需要显示的信息，最后，调用 ShowPhoto 方法显示头像信息。

(8) ShowPhoto 方法用于显示用户头像，如果用户未设置头像，则根据用户的性别显示一个默认的头像，代码如下：

```
    private void ShowPhoto(int user_id)
    {
        string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
        SqlConnection con = new SqlConnection(strConn);
        con.Open();
        SqlCommand cmd = new SqlCommand("SELECT user_sex,user_photo FROM Z_USER"+
```

```
                " WHERE user_id = " + user_id.ToString(), con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//如果有结果表明记录已存在
            {
                if (!reader.IsDBNull(1))
                    Image1.ImageUrl = "~/touxiang.aspx?userid=" + user_id.ToString();
                else
                    Image1.ImageUrl = reader.GetString(0).Equals("男") ? "~/images/boy.gif" :
"~/images/girl.gif";
            }
            reader.Close();
            cmd = null;
            con.Close();
            con = null;
        }
```

从代码可以看出，显示头像指定 Image 控件的 ImageUrl 为一个 URL 地址，所以还需要添加这个页面。

(9) 通过“添加新项”对话框添加名为 touxiang.aspx 的页面，在后台代码文件中添加 using 语句引入所需的命名空间，然后在页面的 Load 事件中添加如下代码：

```
        protected void Page_Load(object sender, EventArgs e)
        {
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT user_photo FROM Z_USER" +
                " WHERE user_id = " + Request.QueryString["userid"].ToString(), con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//如果有结果表明记录已存在
            {
                if (!reader.IsDBNull(0))
                {
                    byte[] data = reader.GetSqlBytes(0).Buffer;
                    Response.ContentType = "image/gif";
                    Response.OutputStream.Write(data, 0, data.Length);
                }
            }
            reader.Close();
            cmd = null;
            con.Close();
            con = null;
        }
```

因为数据库中存放的用户头像格式是二进制数据，所以从数据库中读取到图像信息后通过指定 ContentType 来指定 touxiang.aspx 传送到客户端的是一个图像数据。

(10) 返回到 Login.aspx 页面，在页面的 Load 事件中判断用户是否已登录，如果未登录则显示 Panel1，隐藏 Panel2，否则显示 Panel2，隐藏 Panel1。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["user_id"] != null)
        ShowInfo(Int32.Parse(Session["user_id"].ToString()));
    else
    {
        Panel1.Visible = true;
        Panel2.Visible = false;
    }
}
```

(11) 编译并运行程序，在浏览器中打开登录页面，如图 5-26 所示。输入正确的登录名和密码后，将显示用户的相关信息，如图 5-27 所示。

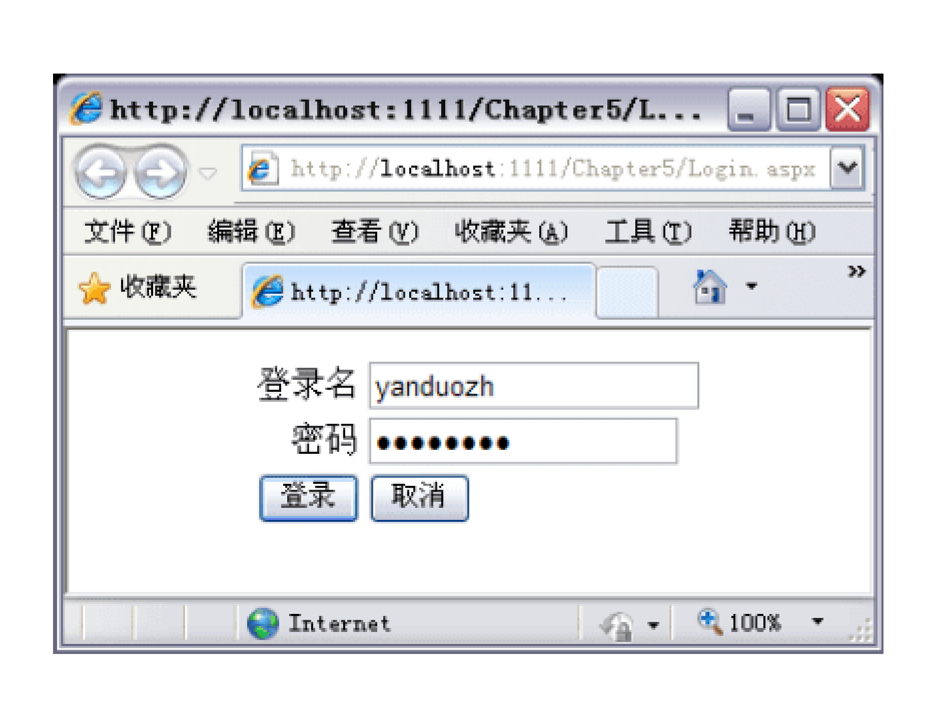


图 5-26　登录页面



图 5-27　登录成功页面

3. 维护个人信息

例 5-8：对于登录成功的用户，提供修改个人信息的功能。包括上传头像，修改除登录名、注册时间和个人微博地址之外的其他信息。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”对话框添加一个名为 Modify.aspx 的页面。

(3) 用户访问修改个人信息页面时，应该是已经登录成功后，所以该页首先要显示当前登录用户的原有信息。通过表格布局方式布局页面，添加相应的控件显示用户信息，页面设计如图 5-28 所示。

(4) 切换到页面的后台代码文件，添加 using 语句引入访问数据库所需的命名空间。

(5) 在页面的 Load 事件中添加代码，加载用户的已有信息并显示。代码如下：

```
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                Calendar1.Visible = false;
                if (Session["user_id"] != null)
                    ShowInfo(Int32.Parse(Session["user_id"].ToString()));
                else
                {
                    Response.Redirect("~/Login.aspx");
                }
            }
        }
        private void ShowInfo(int user_id)
        {
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT
user_name,user_email,user_address,user_info,user_birthday" +
                ",user_telephone,user_sex,user_photo FROM Z_USER WHERE user_id = " +
user_id.ToString(), con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//如果有结果表明记录已存在
            {
                TextBoxName.Text = reader.GetString(0);
                if (!reader.IsDBNull(1))
                    TextBoxEmail.Text = reader.GetString(1);
                else
                    TextBoxEmail.Text = "";
                if (!reader.IsDBNull(2))
                    TextBoxAddress.Text = reader.GetString(2);
                else
                    TextBoxAddress.Text = "";
                if (!reader.IsDBNull(3))
                    TextBoxInfo.Text = reader.GetString(3);
                else
                    TextBoxInfo.Text = "";
                if (!reader.IsDBNull(4))
                    TextBoxBirthday.Text = reader.GetDateTime(4).ToString();
                else
                    TextBoxInfo.Text = "";
                if (!reader.IsDBNull(5))
```

```
                TextBoxTelephone.Text = reader.GetString(5);
            else
                TextBoxTelephone.Text = "";
            RadioButtonList1.SelectedValue = reader.GetString(6);
            if (!reader.IsDBNull(7))
                Image1.ImageUrl = "~/touxiang.aspx?userid=" + user_id.ToString();
            else
                Image1.ImageUrl = reader.GetString(6).Equals("男") ? "~/images/boy.gif" :
"~/images/girl.gif";
        }
        reader.Close();
        cmd = null;
        con.Close();
        con = null;
    }
```

(6) 用于选择生日的 Calendar 控件默认是不可见的，当单击“打开日历”链接按钮时，将显示 Calendar 控件供用户选择日期，选择生日后，将选定日期保存到“生日”文本框中同时隐藏 Calendar 控件。相应的代码如下：

```
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        Calendar1.Visible = true;
    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        TextBoxBirthday.Text = Calendar1.SelectedDate.ToString();
        if(! (TextBoxBirthday.Text==""))
            Calendar1.Visible = false;
    }
```

(7) 修改个人资料需要进行密码验证，如果验证通过则把用户提交的新数据保存至数据库，添加“提交”按钮的单击事件处理程序，代码如下：

```
    protected void ButtonModify_Click(object sender, EventArgs e)
    {
        if (Session["user_id"] != null)
        {
            if (TextBoxName.Text == "")
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"
姓名不能为空\"); ", true);
                return;
            }
```

```
if(CheckPassword(Int32.Parse(Session["user_id"].ToString()),TextBoxPassword.Text))
                    ModifyInfo(Int32.Parse(Session["user_id"].ToString()));
                else
                    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"
密码错误,不允许修改\"); ", true);
            }
            else
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"会话
已过期，请重新登录\"); ", true);
                Response.Redirect("~/Login.aspx");
            }
        }
        private bool CheckPassword(int user_id,string pass)
        {
            bool ret = false;
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT user_name FROM Z_USER WHERE
user_id = "
                + user_id.ToString() +" and user_password='" + pass +"'", con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//如果有结果表明验证通过
            {
                ret =true;
            }
            cmd = null;
            con.Close();
            con = null;
            return ret;
        }
        private void ModifyInfo(int user_id)
        {
            string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConn);
            con.Open();
            SqlTransaction tran = con.BeginTransaction();
            SqlCommand cmd = new SqlCommand("UPDATE   Z_USER SET user_name = @name"
                +" ,user_sex=@sex ,user_email=@email ,user_address=@address ,user_info=@info"
                +" ,user_telephone=@telephone ,user_birthday=@birthday WHERE user_id=" +
user_id.ToString(), con);
```

```
            cmd.Transaction = tran;
            cmd.Parameters.AddWithValue("@name", TextBoxName.Text);
            cmd.Parameters.AddWithValue("@sex", RadioButtonList1.SelectedValue);
            cmd.Parameters.AddWithValue("@email", TextBoxEmail.Text);
            cmd.Parameters.AddWithValue("@address", TextBoxAddress.Text);
            cmd.Parameters.AddWithValue("@info", TextBoxInfo.Text);
            cmd.Parameters.AddWithValue("@telephone", TextBoxTelephone.Text);
            if(TextBoxBirthday.Text=="")//如果没设置生日,需要使用 DBNull
                cmd.Parameters.AddWithValue("@birthday", DBNull.Value);
            else
                cmd.Parameters.AddWithValue("@birthday", TextBoxBirthday.Text);
            try
            {
                int count = cmd.ExecuteNonQuery();
                if (FileUpload1.HasFile)
                {
                    byte[] data = new byte[FileUpload1.FileContent.Length + 1];
                    FileUpload1.FileContent.Read(data, 0, (int)FileUpload1.FileContent.Length);
                    cmd.CommandText = "UPDATE   Z_USER SET user_photo = @user_photo
WHERE user_id="
                        + user_id.ToString();
                    cmd.Parameters.AddWithValue("@user_photo", data);
                }
                count = cmd.ExecuteNonQuery();
                tran.Commit();
            }
            catch (Exception ex)
            {
                tran.Rollback();
            }
            finally
            {
                cmd = null;
                con.Close();
                con = null;
            }
            Response.Redirect(Request.RawUrl);
        }
```

说明：

在执行修改操作时，分为两步。首先修改除头像之外的信息，然后判断是否上传了头像，如果上传了就更新头像，这两次更新操作需要放在数据库的一个事物中执行，如果其中一个执行失败，那么就回滚整个事物。

在 ASP.NET 中，可以使用 Connection 和 Transaction 对象开始、提交和回滚事务。具体步骤如下。

- 调用 Connection 对象的 BeginTransaction 方法来标记事务的开始，BeginTransaction 方法返回对 Transaction 的引用。
- 将 Transaction 对象赋给 Command 的 Transaction 属性。
- 执行事务操作。
- 如果事务操作成功，使用 Transaction 对象的 Commit 方法提交事务，否则，使用 Rollback 方法回滚事务。

(8) 打开例 5-7 中的登录页面，在 Panel2 面板中显示头像的 Image 控件旁边添加一个超链接控件，用于指向 Modify.aspx 页面，如图 5-29 所示。



图 5-28　页面布局



图 5-29　添加超链接控件

(9) 编译并运行程序，在浏览器中打开 Modify.aspx 页面，由于还没有登录，所以页面将跳转到 Login.aspx，登录成功后，单击“维护个人信息”链接进入 Modify.aspx 页面，如图 5-30 所示。

(10) 直接编辑要修改的项，如果要上传头像，则单击“浏览”按钮，选择头像文件，最后输入密码，单击“提交”按钮，完成修改操作，如图 5-31 所示。



图 5-30　Modify.aspx 页面效果



图 5-31　修改个人信息

5.4　数据绑定与数据控件

在 5.3 节介绍了如何通过 ADO.NET 访问数据库，在读取到数据后，只是通过简单的控件来显示这些数据，当数据比较多时，这种方式显然难以处理。下面将学习如何利用 ASP.NET 提供的控件来呈现数据。首先介绍单值绑定和列表控件的数据绑定过程，然后介绍 GridView 等复杂数据绑定控件的基本用法。

5.4.1　数据绑定概述

Web 系统的一个典型的特征是后台对数据的访问和处理与前台数据的显示分离，而前台显示是通过 HTML 来实现的。一种将数据呈现的最直接的方式是将需要显示的数据和 HTML 标记拼接成字符串并输出，但这种方案的缺点也是显而易见的，不但复杂而且难以重用，尤其是有大宗数据需要处理时。为了简化开发过程，ASP.NET 环境中提供了多种不同的服务器端控件来帮助程序员快速高效地完成数据的呈现。

这些用于数据呈现的 ASP.NET 控件，集成了常见的数据显示框架和数据处理功能，在使用时只需设置某些属性，并将需要显示的数据交付给控件，控件就可以帮助用户按照固定的样式(例如表格)，或通过模板自定义样式将一系列数据呈现出来，同时还自动继承某些内置的数据处理功能，如排序和分页等。这些控件被称为数据绑定控件，而将数据交付给数据绑定控件的过程就称为数据绑定。

数据绑定控件本质上依然是通过 HTML 来呈现数据，只不过按照某种样式生成 HTML 框架并将数据填入其中的工作由控件自动完成了，一些复杂的数据绑定控件还提供了大量的功能帮助用户对数据进行进一步操作，如排序、过滤、新增、修改和删除等，因而使得数据呈现的过程变得简单而灵活。

5.4.2　单值绑定和多值绑定

数据绑定实际上是在 HTML 标记中或服务器控件中设置要显示数据的过程。对于页面中的 HTML 标记，可以通过直接嵌入数据或绑定表达式来设置要显示的数据，而对于服务器控件来说，通常通过设置控件属性或指定数据源来完成数据的绑定，并控制其呈现的样式。常用的绑定表达式具有以下形式：<%#XXX%>，绑定表达式可以直接嵌入到前台页面代码中去，通常用于 HTML 标记中的数据显示或单值控件数据设置，如 Label、TextBox 等。而对于列表控件(如 DropDownList、CheckBoxList)，以及后面要着重介绍的复杂数据绑定控件则常采用设置数据源的方式完成数据呈现。

1. 单值绑定

单值绑定其实就是实现动态文本的一种方式。为了实现单值绑定，可影响页面中添加一些特殊的绑定表达式。主要有以下几种数据绑定表达式。

- <%= XXX %>：内联引用方式，可以引用 C#代码。
- <%# XXX %>：可以引用.cs 文件中的字段，但该字段必须初始化后，初始化工作可在页面的 Load 事件中使用 Page.DataBind 的方法来实现。
- <% $XXX %>：可以引用 web.config 文件中预定义的字段或者已注册的类。
- <%# Eval(XXX) %>：当一个 ASP.NET 控件位于一个数据绑定模板中时，可以使用 Eval()方法将其某个属性与数据源中当前数据对象的某个属性相绑定。Eval 方法提供了一个单向的只读的数据值。这就是说，数据是从“数据源”对象单向传送给模板中的控件，且没有办法修改数据源对象中的数据。
- <%# Bind(XXX) %>：当需要修改数据源中的数据时，通常采用 Bind()方法实现这一功能。

例如，下面的数据绑定表达式都是有效的：

```
<%# DateTime.Now %>
<%# 3+(6*number) %>    //其中，number 是 Web 页后置代码类中的 public 或 protected 变量
<% $connectionStrings:WeiBoConnectionString %>    //引用 web.config 中的数据库连接字符串
```

单值绑定有如下两个缺点。

(1) 数据绑定代码与定义用户界面的代码混合在一起。

(2) 代码过于分散。正是由于这两个缺点才造成不方便管理页面和代码，所以应尽量少用单值绑定。

2. 多值绑定

多值绑定通常与列表控件以及复杂的数据控件一起工作，可以把多条数据一次绑定到这些控件中。

多值绑定的步骤如下。

(1) 把存储数据的数据对象绑定到列表控件或数据控件的 DataSource 属性。

(2) 调用控件或者当前页面的 DataBind()方法。

为了创建多值绑定，需要使用支持数据绑定的控件，ASP.NET 提供了一系列这类控件，这些控件如下。

- 列表控件：ListBox、DropDownList、CheckBoxList 和 RadioButtonList 等。
- HtmlSelect 控件：这是一个 HTML 控件，类似于 ListBox 控件。
- 复杂数据控件：GridView、DetailsView、FormView 和 ListView 等。

5.4.3　数据控件简介

为了有效处理系统中的数据，ASP.NET 工具箱的“数据”类别中提供了两组数据感知控件，数据源控件和数据绑定控件。数据源控件用于从数据源(如数据库或 XML 文件)中检索数据，然后将这一数据提供给数据绑定控件；数据绑定控件可用于显示和编辑数据。

1. 数据源控件

数据源控件用于连接数据源、从数据源中读取数据以及把数据写入数据源。数据源控件不呈现任何用户界面，而是充当特定数据源与 ASP.NET 网页上的其他控件之间的桥梁。数据源控件实现了各种数据检索和修改功能，其中包括查询、排序、分页、筛选、更新、删除以及插入等。

在工具箱的“数据”类别下包含了 7 个不同的数据源控件和 1 个增件。

- ObjectDataSource 控件：具有数据检索和更新功能的中间层对象，允许使用业务对象或其他类，并可创建依赖中间层对象管理数据的 Web 应用程序。
- SqlDataSource 控件：用来访问存储在关系数据库中的数据源，包括 Microsoft SQL Server、Oracle、OLE DB 和 ODBC。当该控件与 SQL Server 一起使用时支持高级缓存功能；当数据源作为 DataSet 对象返回时，该控件还支持排序、筛选和分页功能。
- AccessDataSource 控件：用于在 Web 页面中显示 Microsoft Access 数据库的数据。它非常简单，在某种程度上类似于 SqlDataSource 控件，因为它允许处理来自数据库的数据。但不同之处在于它只针对 Microsoft Access 数据库进行优化。
- XmlDataSource 控件：主要用于绑定分层的、基于 XML 的数据，该控件支持使用 XPath 表达式来实现筛选功能，并允许对数据应用 XSLT 转换，此外，还允许通过保存更改后的 XML 文档来更新数据。
- SiteMapDataSource 控件：该控件结合导航控件实现站点导航功能，第 3 章在介绍导航控件时曾介绍过如何使用 SiteMapDataSource。
- EntityDataSource 控件：EntityDataSource 控件之于 EF(Entity Framework)就像 SqlDataSource 控件之于基于 SQL 的数据源，它提供了一个声明性的方法来访问模型。和 SqlDataSource 控件一样，EntityDataSource 提供了对 CRUD 操作的轻松访问，另外使数据排序和筛选也变得非常简单。EntityDataSource 通过 LINQ to EF 提供了对底层 SQL Server 数据库的完全访问。
- LinqDataSource 控件：用作 LINQ to SQL 的数据源。LINQ to SQL 是一种类似于 EF(将在第 6 章介绍)的技术。因为 Microsoft 现在大力推广的是 EF，而不是 LINQ to SQL，所以本书不讨论该控件。
- QueryExtender 控件：用作 LinqDataSource 和 EntityDataSource 控件的增件，因为它可以用来创建丰富的过滤界面，以便能够搜索特定的数据，而不必手动编写大量代码。

本章将重点介绍 SqlDataSource 控件，SqlDataSource 控件使用 ADO.NET 类与 ADO.NET 支持的任何数据库进行交互。使用该控件，可以在 ASP.NET 页面中访问和操作数据，而无须直接使用 ADO.NET 类，只需提供用于连接到数据库的连接字符串，并定义使用数据的 SQL 语句或存储过程即可。在运行时，SqlDataSource 控件会自动打开数据库连接，执行 SQL 语句或存储过程，返回指定数据，然后关闭连接。

例 5-9：使用数据源控件和数据绑定初始化 CheckBoxList 控件。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”对话框添加一个名为 FindUser.aspx 的页面。

(3) 切换到页面的“设计”视图，添加几个用于输入查询条件的文本框控件，然后添加一个“查询”按钮和一个 CheckBoxList 控件用于显示查询结果，页面布局如图 5-32 所示。

(4) 在工具箱的“数据”类别中双击 SqlDataSource 控件，将添加该控件并打开控件的“任务”面板，单击“配置数据源”选项，如图 5-33 所示。

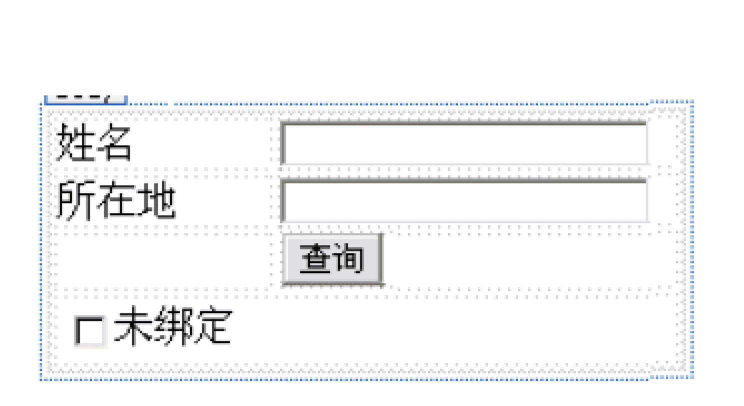


图 5-32　页面布局

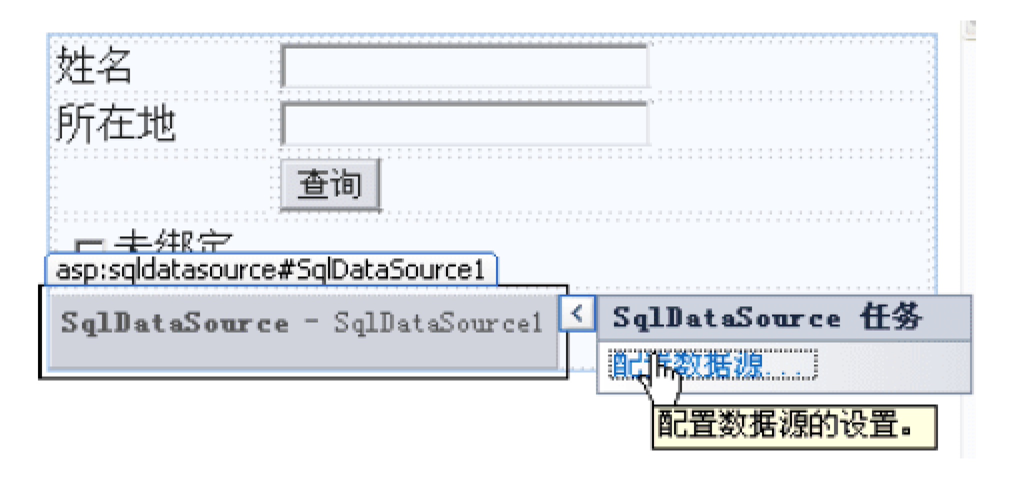


图 5-33　添加 SqlDataSource 控件

(5) 此时将打开“配置数据源”向导对话框，第一步是选择数据连接，因为前面已经添加了数据连接，所以在此可以在下拉列表中直接选择，如果下拉列表中没有，可以单击“新建连接”按钮添加数据连接。这里选择 web.config 已经添加的数据库连接串 WeiBoConnectionString，单击“连接字符串”前面的加号，可以查看该连接生成的连接字符串，如图 5-34 所示。

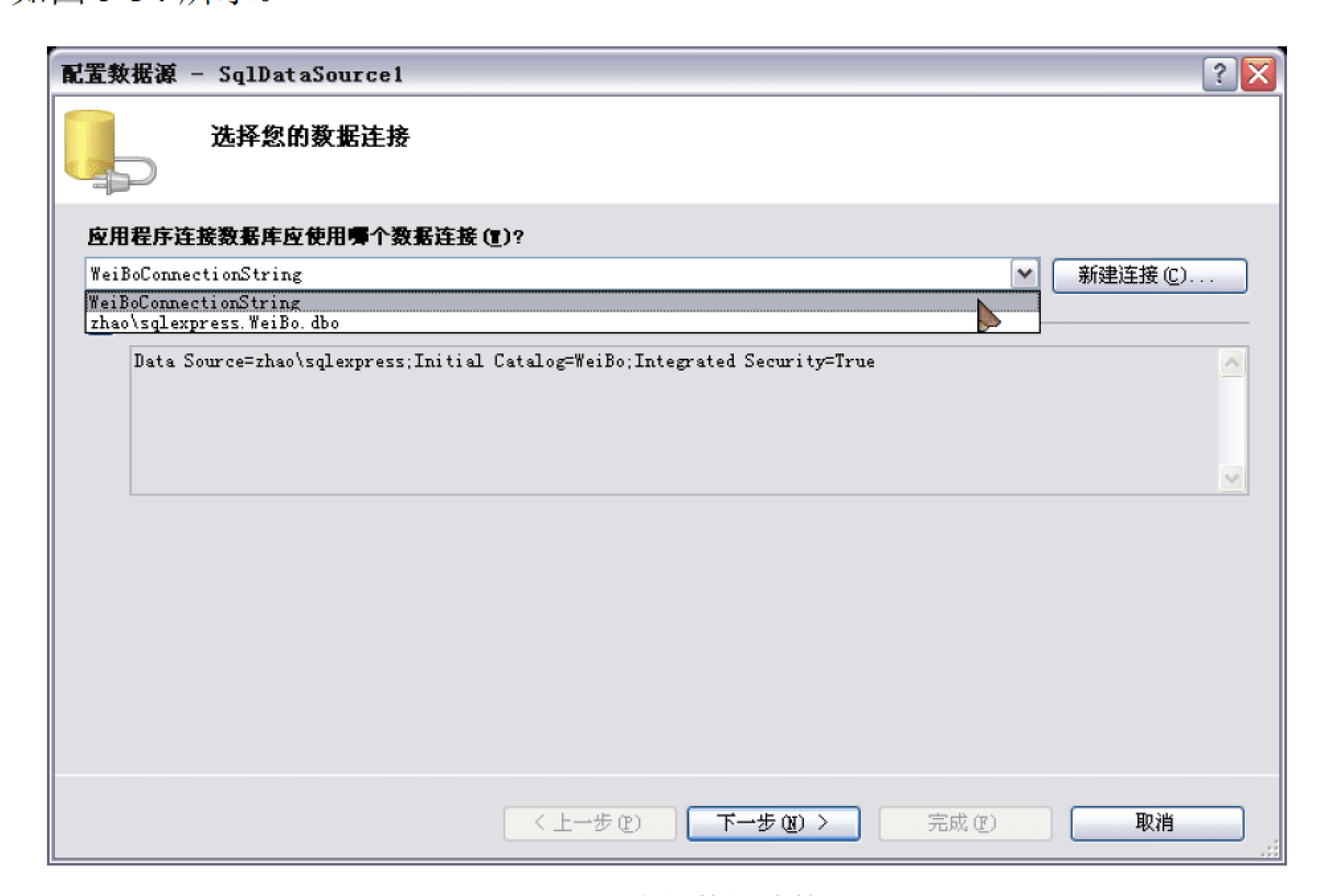


图 5-34　选择数据连接

如果选择的是数据连接，而该连接串还没有保存到 web.config 文件中，则单击“下一步”按钮，将打开如图 5-35 所示的“将连接字符串保存到应用程序配置文件中”对话框。选中“是，将此连接另存为”复选框，即可将其保存到 web.config 文件中。

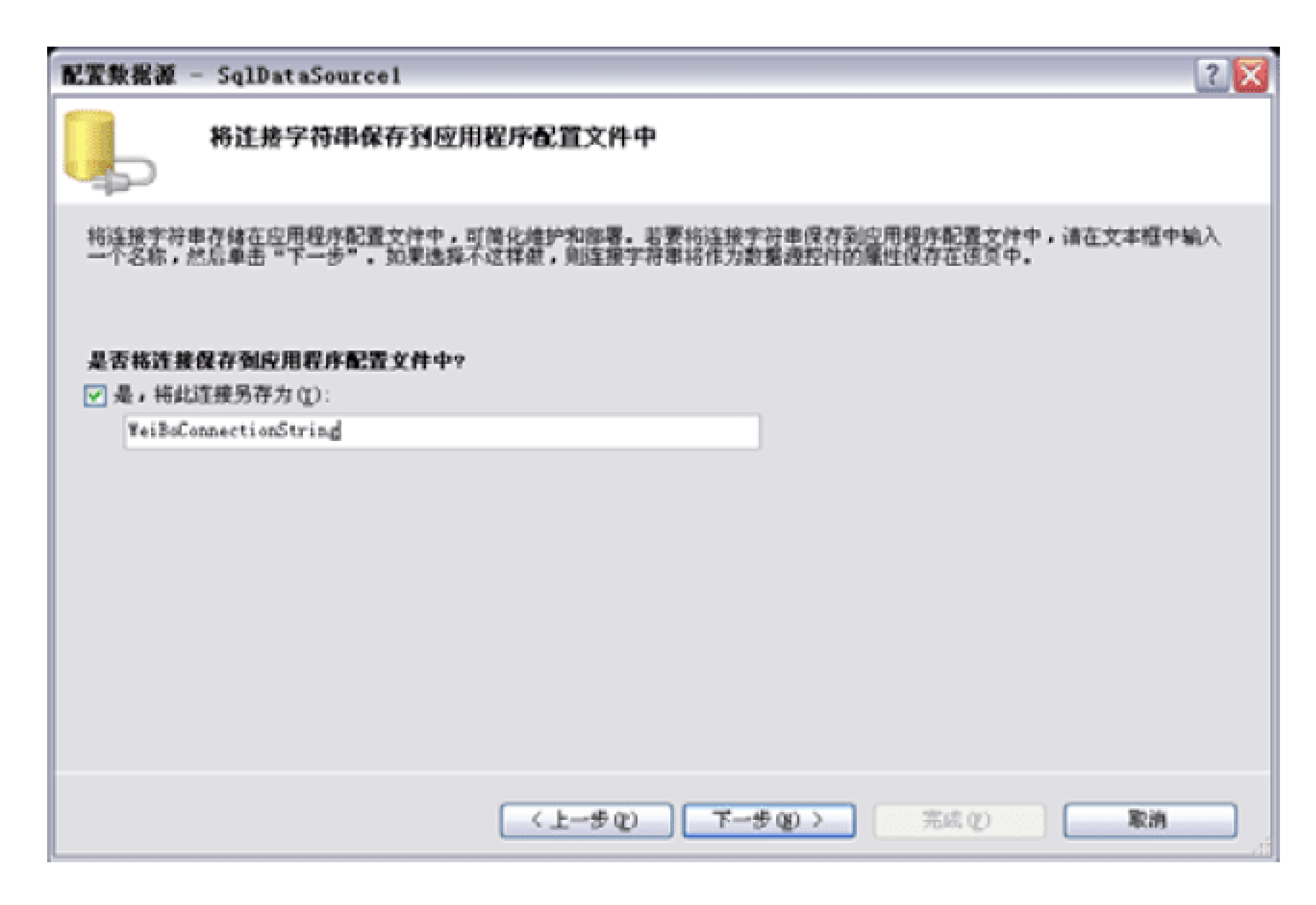


图 5-35　“将连接字符串保存到应用程序配置文件中”对话框

(6) 单击“下一步”按钮，打开如图 5-36 所示的“配置 Select 语句”对话框。在该对话框中可以选择需要的表和列，也可以增加 WHERE 子句和 ORDER BY 子句，还可以单击“高级”按钮进行高级 SQL 生成选项设置。这里选择 Z_USER 表和其中的几个字段。

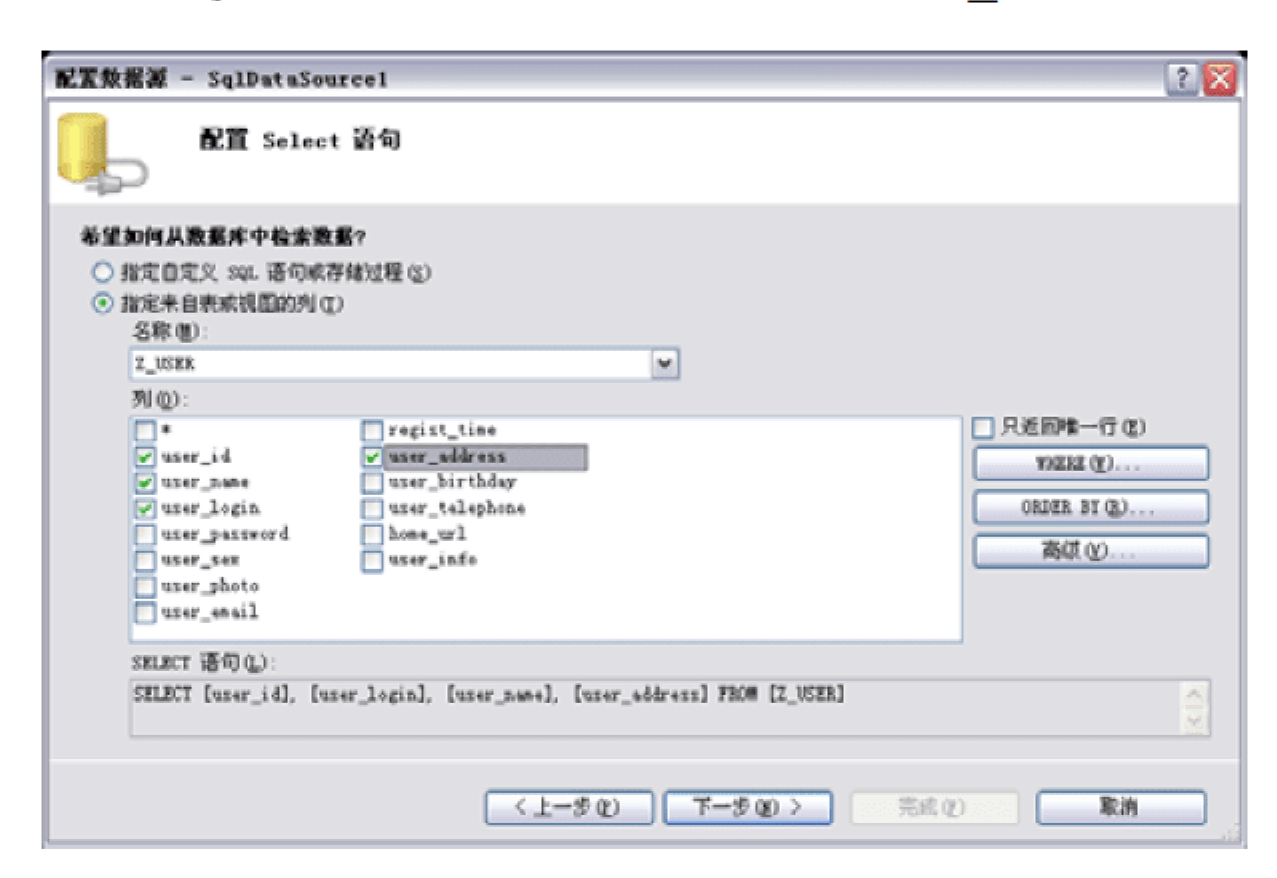


图 5-36　“配置 Select 语句”对话框

(7) 单击 WHERE 按钮，打开如图 5-37 所示的“添加 WHERE 子句”对话框，在此设置查询条件，共设置两个查询条件，分别是 user_name 和 user_address，其值为 LIKE 相应控件的值，如图 5-37 所示。

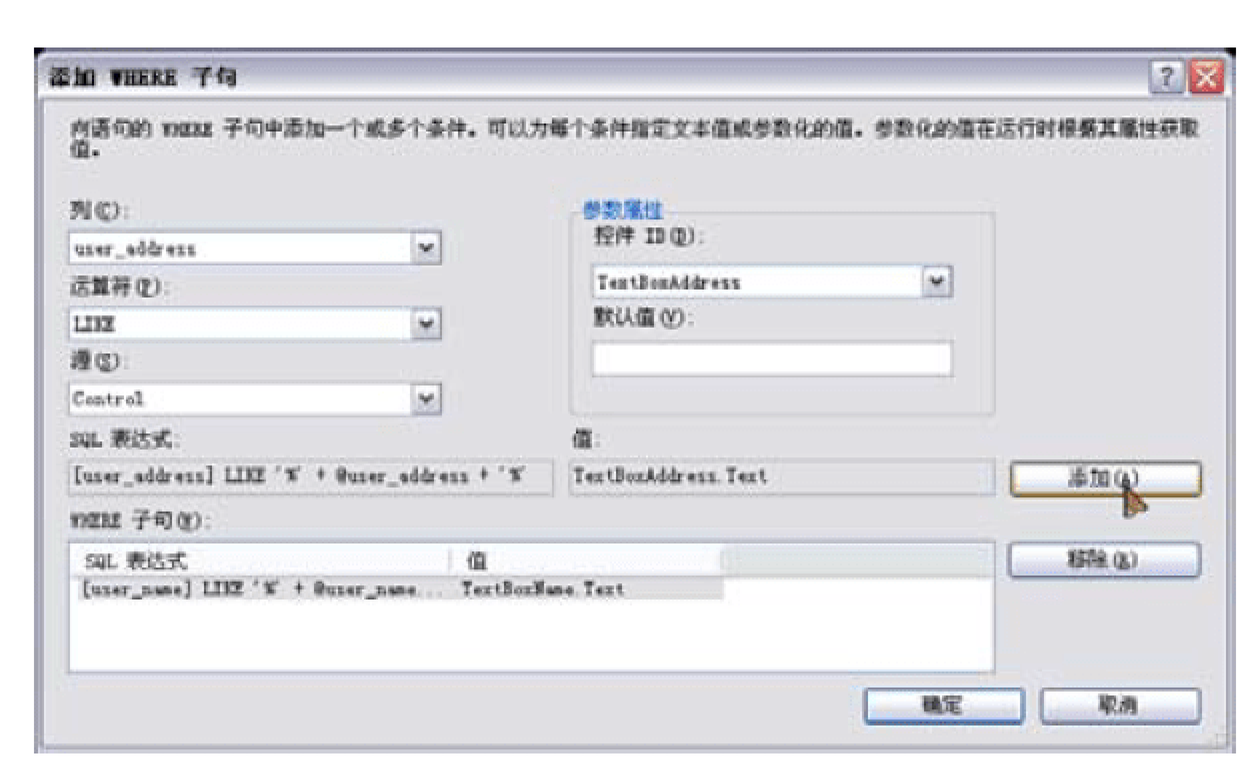


图 5-37　“添加 WHERE 子句”对话框

(8) 添加完两个查询条件后，单击“确定”按钮，返回“配置 Select 语句”对话框，在此对话框中将显示生成的 SELECT 语句，如图 5-38 所示。

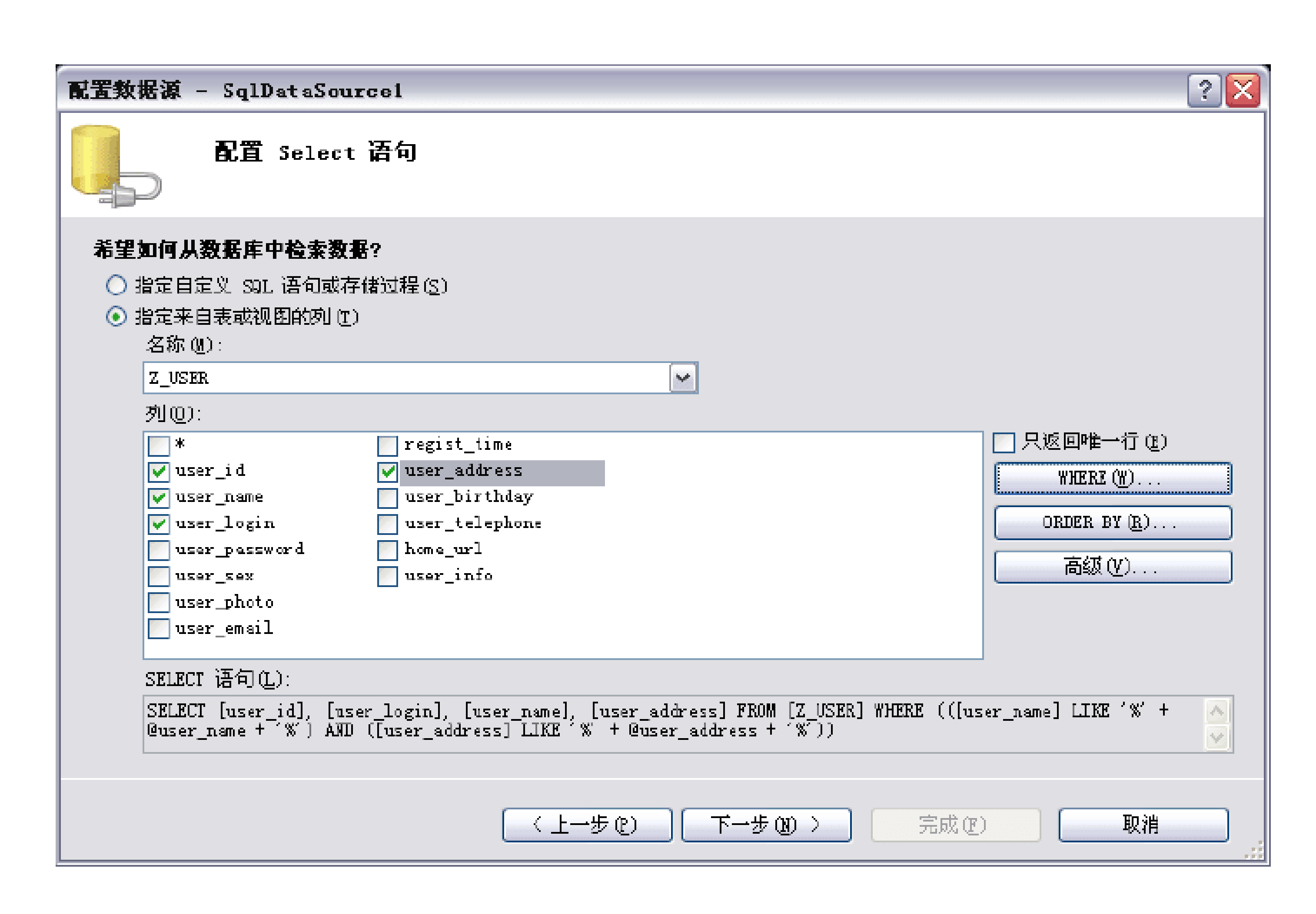


图 5-38　“配置 Select 语句”对话框中显示生成的 SELECT 语句

(9) 单击“下一步”按钮，打开如图 5-39 所示的“测试查询”对话框，单击“测试查询”按钮，将弹出“参数值编辑器”对话框，输入一些查询参数，单击“确定”按钮即可显示查询结果。

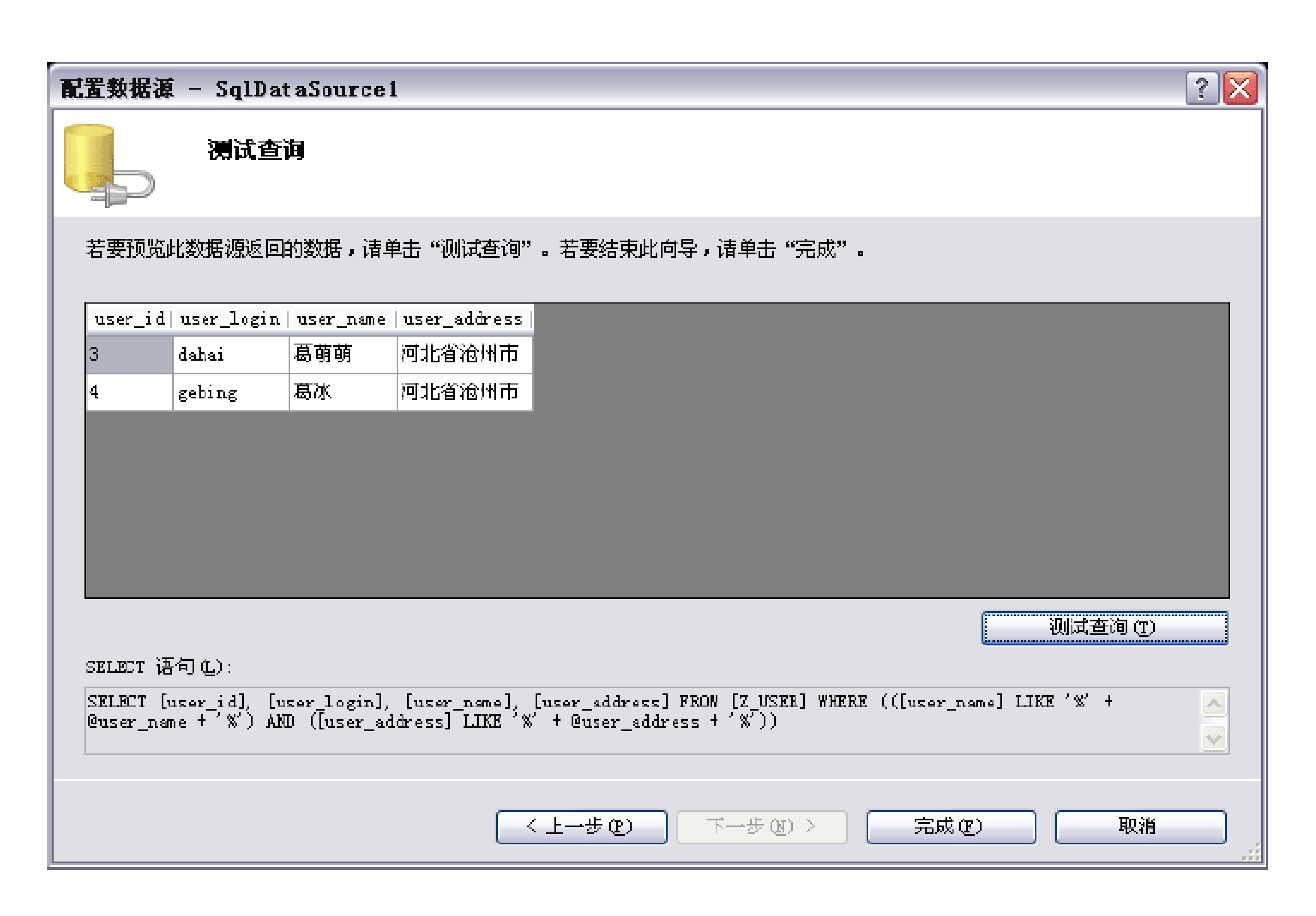


图 5-39　“测试查询”对话框

(10) 单击“完成”按钮，完成数据源的配置。切换到“源”视图，可以看到生成的代码如下所示：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:WeiBoConnectionString %>"
    SelectCommand="SELECT [user_id], [user_login], [user_name], [user_address] FROM
[Z_USER] WHERE (([user_name] LIKE '%' + @user_name + '%') AND ([user_address] LIKE '%' +
@user_address + '%'))">
```

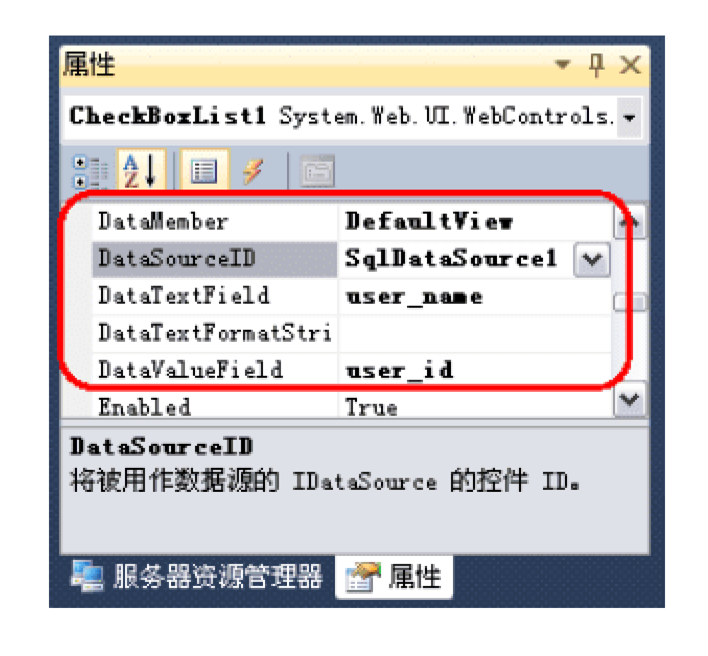
```
        <SelectParameters>
            <asp:ControlParameter ControlID="TextBoxName" Name="user_name"
                PropertyName="Text" Type="String" />
            <asp:ControlParameter ControlID="TextBoxAddress" Name="user_address"
                PropertyName="Text" Type="String" />
        </SelectParameters>
    </asp:SqlDataSource>
```

(11) 在“属性”窗口中设置 CheckBoxList 控件的数据绑定相关的属性，包括 DataSourceID、DataMember、DataTextField 和 DataValueField，如图 5-40 所示。

(12) 双击“查询”按钮，在其单击事件中添加如下代码，实现数据绑定。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        CheckBoxList1.DataBind();
    }
```

(13) 编译并运行程序，在浏览器中打开 FindUser.aspx 页面，输入查询条件，单击“查询”按钮，将显示模糊查询结果，如图 5-41 所示。



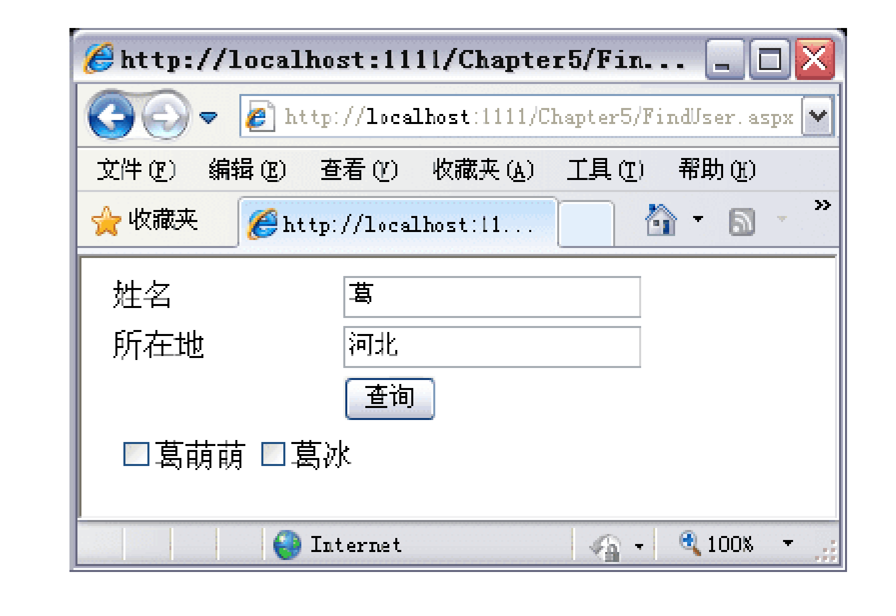



图 5-40　设置数据绑定相关的属性　　　图 5-41　模糊查询结果绑定到 CheckBoxList 控件

2. 数据绑定控件

使用数据绑定控件可以在 Web 页面上显示和编辑数据。在 VWD 2010 工具箱的“数据”类别中有 7 个数据绑定控件。其中，GridView、DataList、ListView 和 Repeater 都可以同时显示多条记录，而 DetailsView 和 FormView 设计为一次显示一条记录，DataPager 是为 ListView 控件提供分页功能的辅助控件。

- GridView 控件：这是一个功能非常多的控件，它支持自动分页(记录被划分到多个“页面”中)、排序、编辑、删除和选择。它像一个带有行和列的电子表格那样呈现数据，其中每行包含一条完整的记录。尽管有许多种可以样式化这些行和控件外观的方法，但不能从根本上改变表现数据的方式。另外，GridView 并不允许直接在底层数据源中插入记录。
- DataList 控件：该控件不仅可以像 GridView 那样以行表现数据记录，也可以以列的形式表现，从而可以创建一种矩阵形式的数据表现方法。另外，它也允许通过一

组模板定义数据的外观。不过，它不支持分页和排序，不允许插入新记录或更新、删除已有记录。

- Repeater 控件：该控件在输出到浏览器的 HTML 方面提供了最大的灵活性，因为该控件本身并不添加任何 HTML 到页面输出中。因此，它常用于 HTML 有序列表或无序列表(<ol>和<ul>)以及其他列表形式。此控件可以通过控件提供的大量模板定义整个客户端标记。不过，它没有分页、排序和修改数据的内置功能。
- ListView 控件：该控件是在 ASP.NET 3.5 中引入的，它最好地合并了 GridView、DataList 和 Repeater。ASP.NET 4 中对 ListView 做出了一些改动，使它更易于使用。类似于 GridView，它支持数据编辑、删除和分页；像 DataList 那样，它支持多列和多行布局，而且还像 Repeater 那样允许完全控制控件生成的标记。
- DetailsView 控件：和 FormView 控件类似，它们都只能一次显示一条记录。DetailsView 使用内置的表格格式显示数据，而 FormView 使用模板来定义数据的外观。
- FormView 控件：ASP.NET 4 中对 FormView 添加了一个新的 RenderOuterTable 属性。当把这个属性设为 True(默认值为 False)时，控件不会生成包装的 HTML <table>元素。这样就会生成更少的代码和更清晰的 HTML。
- DataPager 控件：该控件可以在其他控件上分页。目前，它只能用于扩展 ListView 控件，但随着.NET Framework 未来版本的发布，这一情况将会改善。

3. 其他数据控件

工具箱中“数据”类别中还有一个 Chart 控件。它最初是作为 Visual Studio 2008 的一个增件发布的，但是现在已经完全集成到了 Visual Studio 2010 中。它用来绘制从简单的条形图到 3D 饼图和折线图的各种图形。该控件不在本书讨论范围之内，所以这里不再介绍。

5.4.4　以主-从表形式显示数据

数据绑定控件已经介绍过了，有的可以同时显示多条记录，而有的一次只显示一条记录，下面就使用这两类控件以主-从表形式显示数据。

例 5-10：使用 GridView 和 DetailsView 控件以主-从表的形式显示信息表 Z_MESSAGE 和用户表 Z_USER 中的数据。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”对话框添加一个名为 ShowMsg.aspx 的页面。

(3) 切换到页面的“设计”视图，添加一个 GridView 到窗体中，将自动打开“GridView 任务”面板，从“选择数据源”下拉列表中选择“<新建数据源>”选项，如图 5-42 所示。

(4) 在打开的“数据源配置向导”对话框中选择数据源类型为“SQL 数据库”，如图 5-43 所示，单击“确定”按钮。

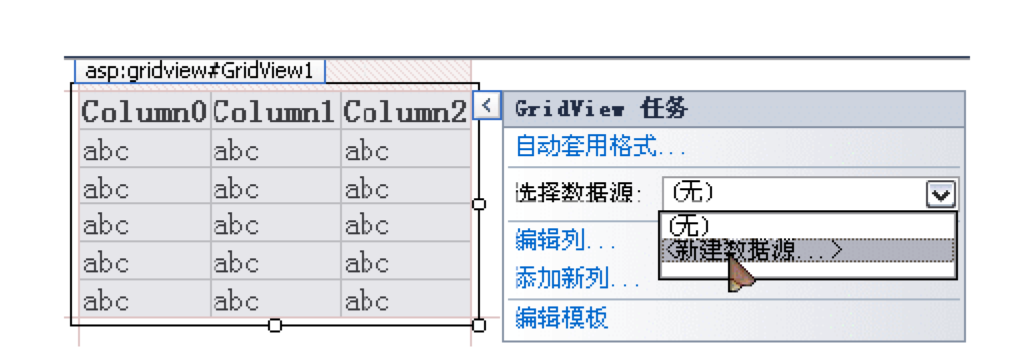


图 5-42　添加 GridView 控件并配置数据源

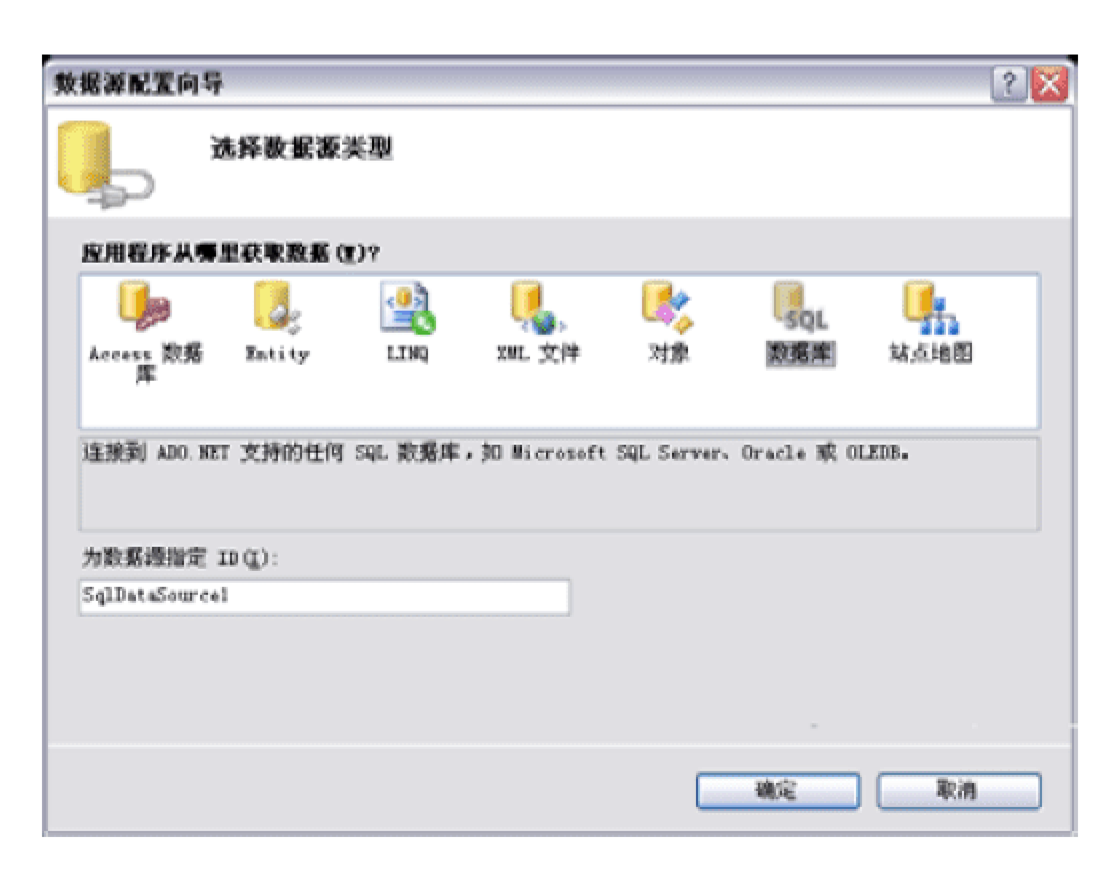


图 5-43　“数据源配置向导”对话框

接下来的步骤与例 5-9 中创建数据源的方法类似，不同的是，这里选择的表是 Z_MESSAGE 表，字段选择“*”，即表中的所有列。生成的数据源代码如下所示：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:WeiBoConnectionString %>"
    SelectCommand="SELECT * FROM [Z_MESSAGE]"></asp:SqlDataSource>
```

(5) 在“GridView 任务”面板中选择“编辑列”选项，打开“字段”对话框，可以在该对话框中设置 GridView 控件显示的字段。在“选定的字段”列表中，将每个字段的 HeaderText 属性都改为中文，然后，在“可用字段”中将 CommandField 下面的命令按钮通过“添加”按钮，添加到选定的字段中，如图 5-44 所示。

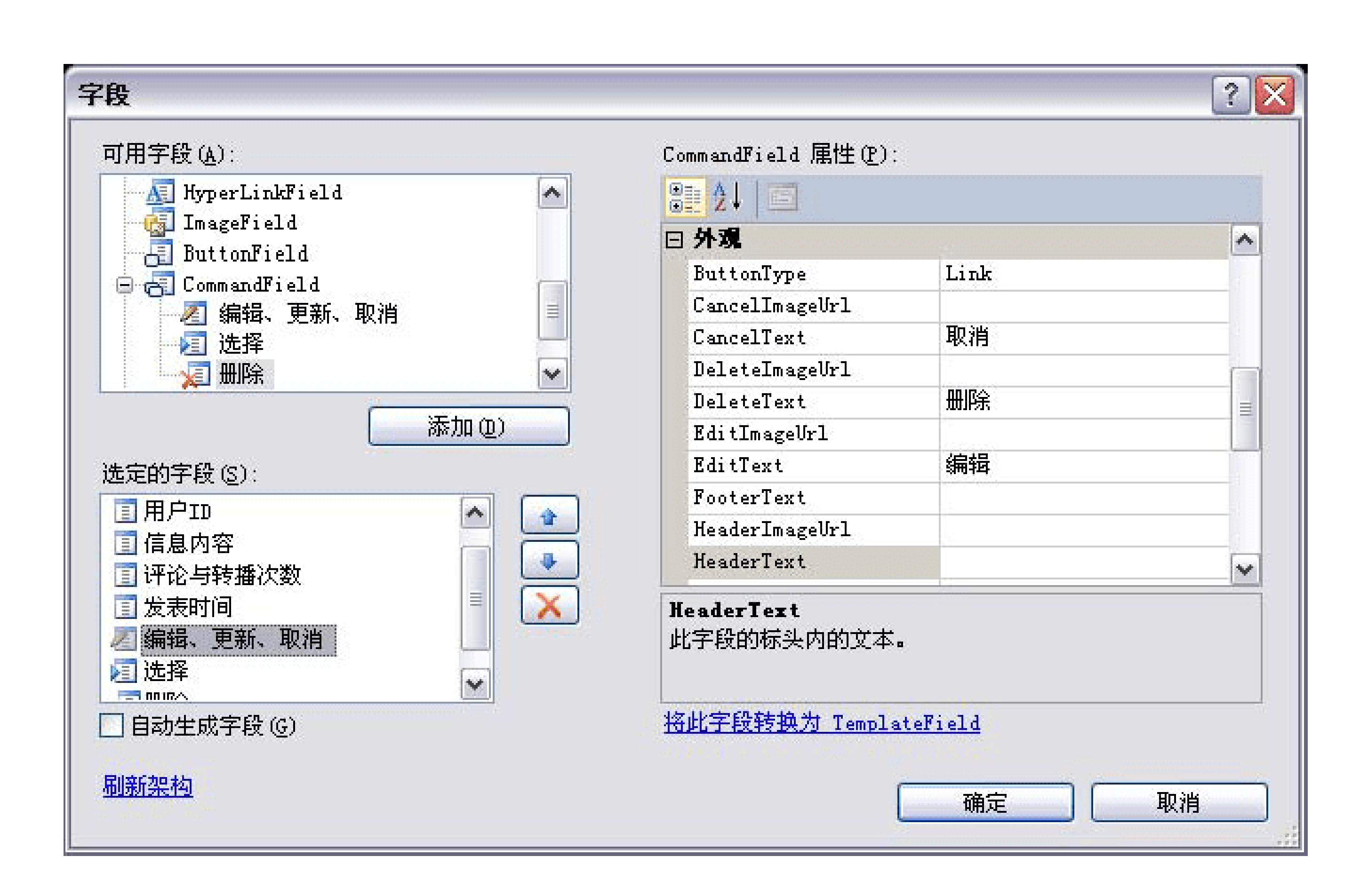


图 5-44　“字段”对话框

(6) 选中“GridView 任务”面板中的“启用分页”和“启用排序”复选框，在“属性”面板中设置控件的 DataKeyNames 属性为 msg_id,user_id。

(7) 通过“GridView 任务”面板中的“自动套用格式”选项为控件指定一个好看的外观。

(8) 在 GridView 控件的下方添加一个 DetailsView 控件，为 DetailsView 控件新建数据源，在“配置 Select 语句”一步中选项 Z_USER 表中除用户密码和用户头像之外的其他字

段，然后单击 WHERE 按钮，打开“添加 WHERE 子句”对话框，设置 user_id 列的值为控件 GridView1，如图 5-45 所示，生成的数据源代码如下：

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:WeiBoConnectionString %>"
    SelectCommand="SELECT [user_id], [user_name], [user_login], [user_sex],
[user_email], [user_info], [home_url], [user_telephone], [user_birthday], [user_address], [regist_time]
FROM [Z_USER] WHERE ([user_id] = @user_id)">
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="user_id"
            PropertyName="SelectedValue" Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

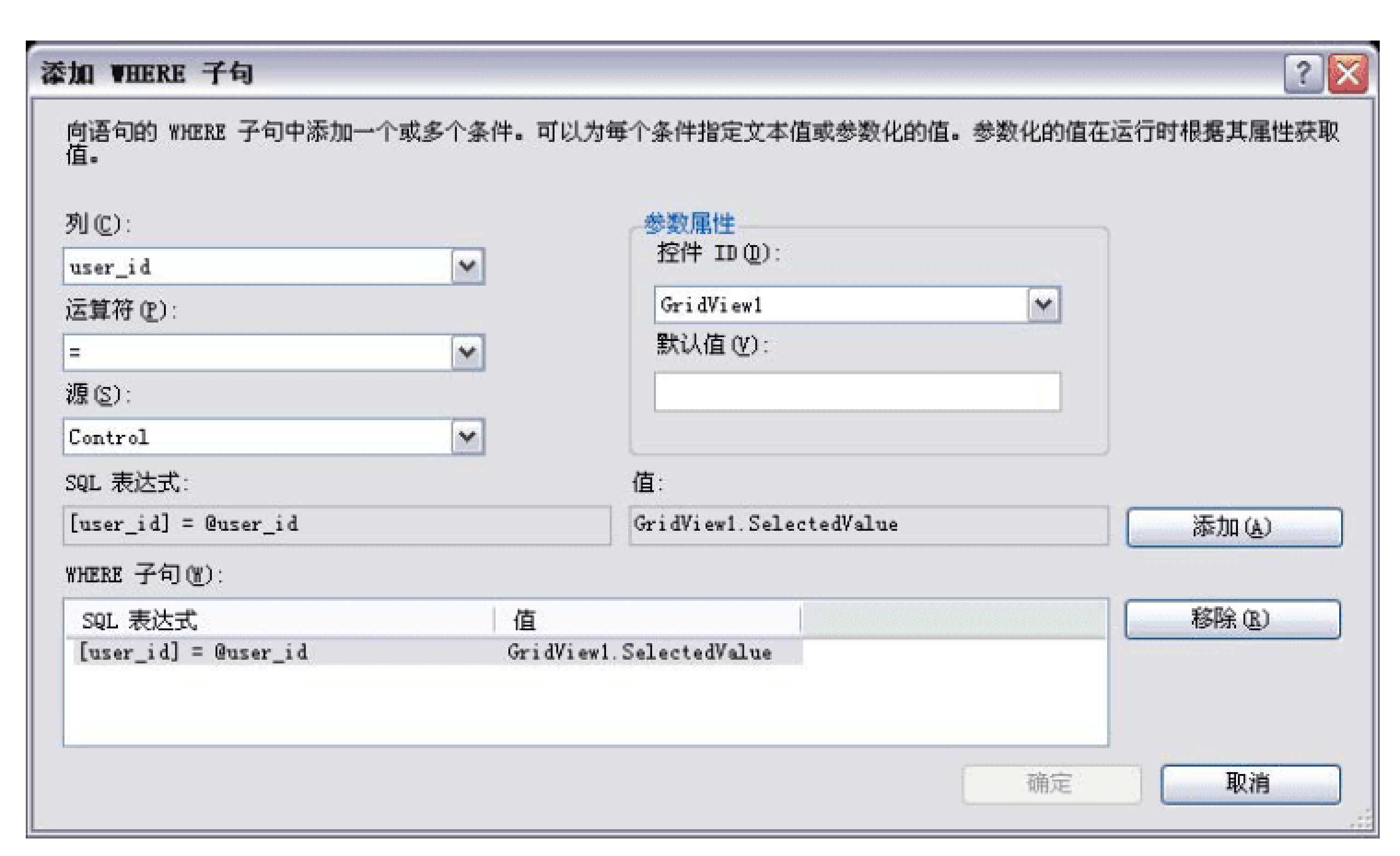


图 5-45　“添加 WHERE 子句”对话框

(9) 上述代码中 GridView 控件的 SelectedValue 属性对应 GridView 控件的 DataKeyNames 字段的值，由于前面设置 GridView 控件的 DataKeyNames 为两个字段，所以此处略作修改，将<SelectParameters>修改为如下代码：

```
<SelectParameters>
    <asp:ControlParameter ControlID="GridView1" Name="user_id"
        PropertyName="SelectedDataKey.Values[1]" Type="Int32" />
</SelectParameters>
```

(10) 通过“DetailsView 任务”面板中的“编辑字段”选项，将控件中的字段信息都修改为中文，通过“自动套用格式”选项设置控件的外观。

(11) 至此，已完成所有工作，没有编写任何代码，一个简单的主-从数据显示页面就做好了，编译并运行程序，在浏览器中打开 ShowMsg.aspx 页面。当选择 GridView 控件中的某一条记录时，下面将显示该信息的作者信息，如图 5-46 所示。

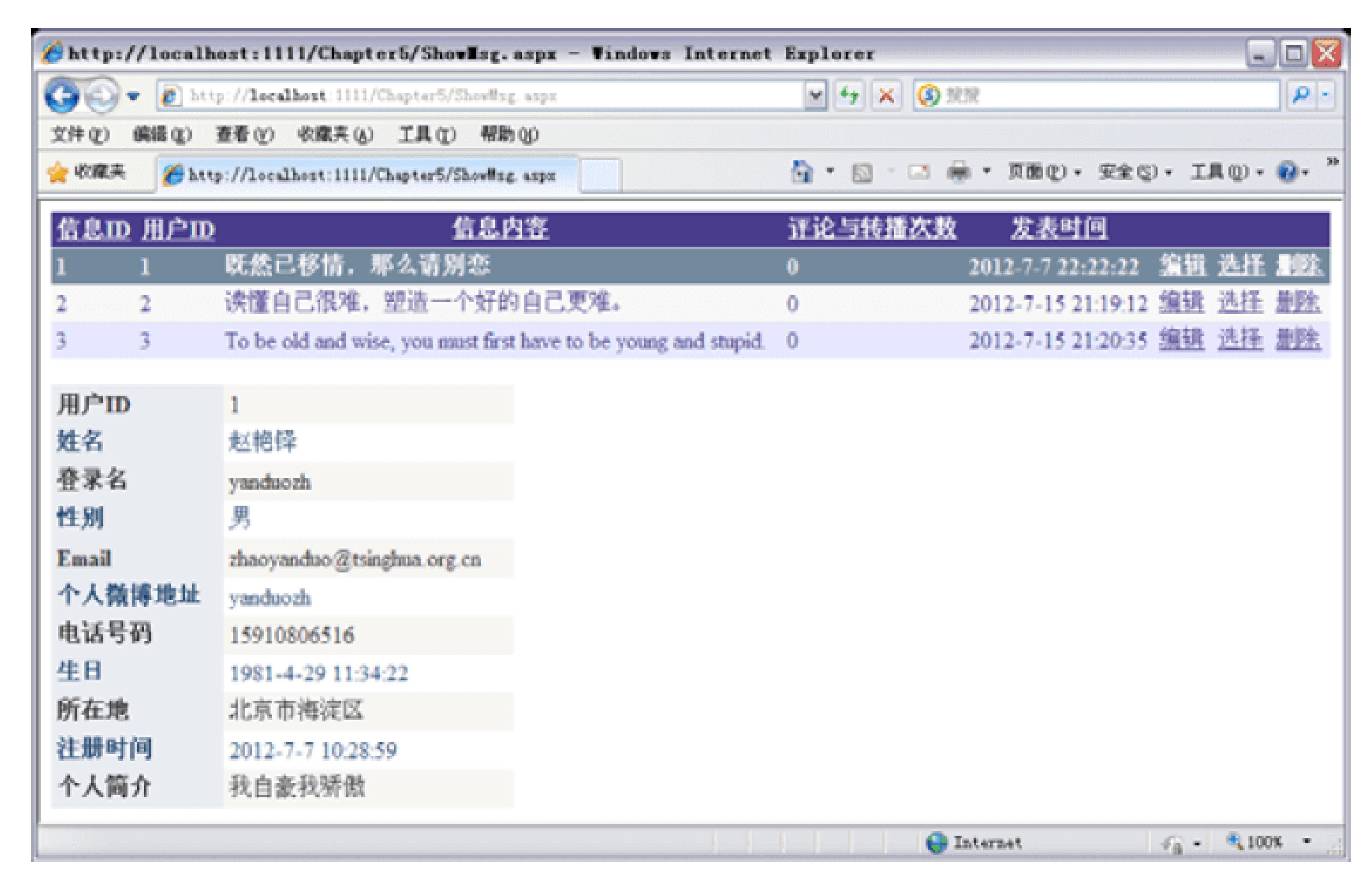


图 5-46　主-从信息表显示

5.5　使用 ADO.NET 访问 XML

如何使用 ADO.NET 访问数据库的问题已经讨论过了。数据库是进行数据存储和管理的一种习惯的方式。现在，XML 已逐步成为数据存储的一种新的方式，因此可以考虑将数据保存在 XML 文档中，并采用一定的方法对它进行管理。ADO.NET 提供了对 XML 数据访问的功能。

5.5.1　XML 概述

XML(eXtensible Markup Language，称为可扩展标记语言)是一种可以用来创建自己的标记的标记语言。它由万维网协会(W3C)创建，用来克服 HTML(即超文本标记语言 Hypertext Markup Language)的局限。和 HTML 一样，XML 基于 SGML——标准通用标记语言(Standard Generalized Markup Language)。XML 是 SGML 上的一个子集，XML 包含了 SGML 的很多特性，但是要比 SGML 简单得多。

XML 是一种类似于 HTML 的标记语言，但是 XML 不是 HTML 的替代品，XML 和 HTML 是两种不同用途的语言，其中最主要的区别是：XML 专门用来描述文本的结构，而不是用于描述如何显示文本的，而 HTML 则是用来描述如何显示文本的。

XML 提供了一种保存数据的格式，数据可以通过这种格式很容易地在不同的应用程序之间实现共享。它是用来存放数据的，可以利用相关的 XML API(MSXML DOM、JAVA DOM 等)对 XML 进行存取和查询。

5.5.2　使用 ADO.NET 读写 XML 数据

使用 DataSet 的 ReadXml 方法可以读取所有 XML 文档数据，使用 WriteXml 方法可以将数据保存到 XML 文件中。下面通过一个例子来说明读写 XML 文件的方法。

例 5-11：使用 DataSet 读写 XML 文档。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”命令，添加一个名为 Users.xml 的 XML 文件，文件的内容如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<Users>
  <User>
    <user_id>1</user_id>
    <user_name>赵艳铎</user_name>
    <user_sex>男</user_sex>
    <user_telephone>15910806516</user_telephone>
    <user_address>北京市海淀区</user_address>
    <user_info>世上没有永不分离</user_info>
  </User>
  <User>
    <user_id>2</user_id>
    <user_name>小石头</user_name>
    <user_sex>男</user_sex>
    <user_telephone>82166054</user_telephone>
    <user_address>浙江省杭州市</user_address>
    <user_info>万念俱灰，踌躇满志</user_info>
</User>
  <User>
    <user_id>3</user_id>
    <user_name>葛萌萌</user_name>
    <user_sex>女</user_sex>
    <user_telephone>03172053456</user_telephone>
    <user_address>河北省沧州市</user_address>
    <user_info>你也许已走出我的视线，但从未走出我的思念</user_info>
  </User>
</Users>
```

(3) 通过“添加新项”命令，添加一个名为 XmlTest.aspx 的页面，打开页面的“设计”视图，添加两个 Button 控件和一个 GridView 控件，Button 控件的 Text 属性分别为“加载数据”和“修改并保存数据”。

(4) 双击按钮，为两个按钮添加单击事件处理程序，代码如下：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
```

```
            DataSet ds = new DataSet();
            ds.ReadXml(Server.MapPath("Users.xml"));//读取 XML 数据到 DataSet 数据集
            GridView1.DataSource = ds.Tables[0].DefaultView;//绑定数据源
            GridView1.DataBind();
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            DataSet ds = new DataSet();
            ds.ReadXml(Server.MapPath("Users.xml"));
            DataTable dtable = ds.Tables[0];
            //用 DataRowCollection 对象获取这个数据表的所有数据行
            DataRowCollection rows = dtable.Rows;
            //修改操作，逐行遍历，取出各行的数据
            for (int i = 0; i < rows.Count; i++)
            {
                DataRow row = rows[i];
                //修改 user_id 为奇数的 user_info 值
                if(Int32.Parse(row[0].ToString())%2==1)
                    row[5] = "我是"+row[1]+",来自"+row[4];
            }
            ds.WriteXml(Server.MapPath("Users.xml"));//将 DataSet 数据写成 XML 文本
            //绑定数据源，重新加载并显示
            GridView1.DataSource = ds.Tables[0].DefaultView;
            GridView1.DataBind();
        }
```

(5) 编译并运行程序，在浏览器中打开 XmlTest.aspx 文件，单击“加载数据”按钮，读取 xml 文件中的内容，并显示在 GridView 控件中，如图 5-47 所示。

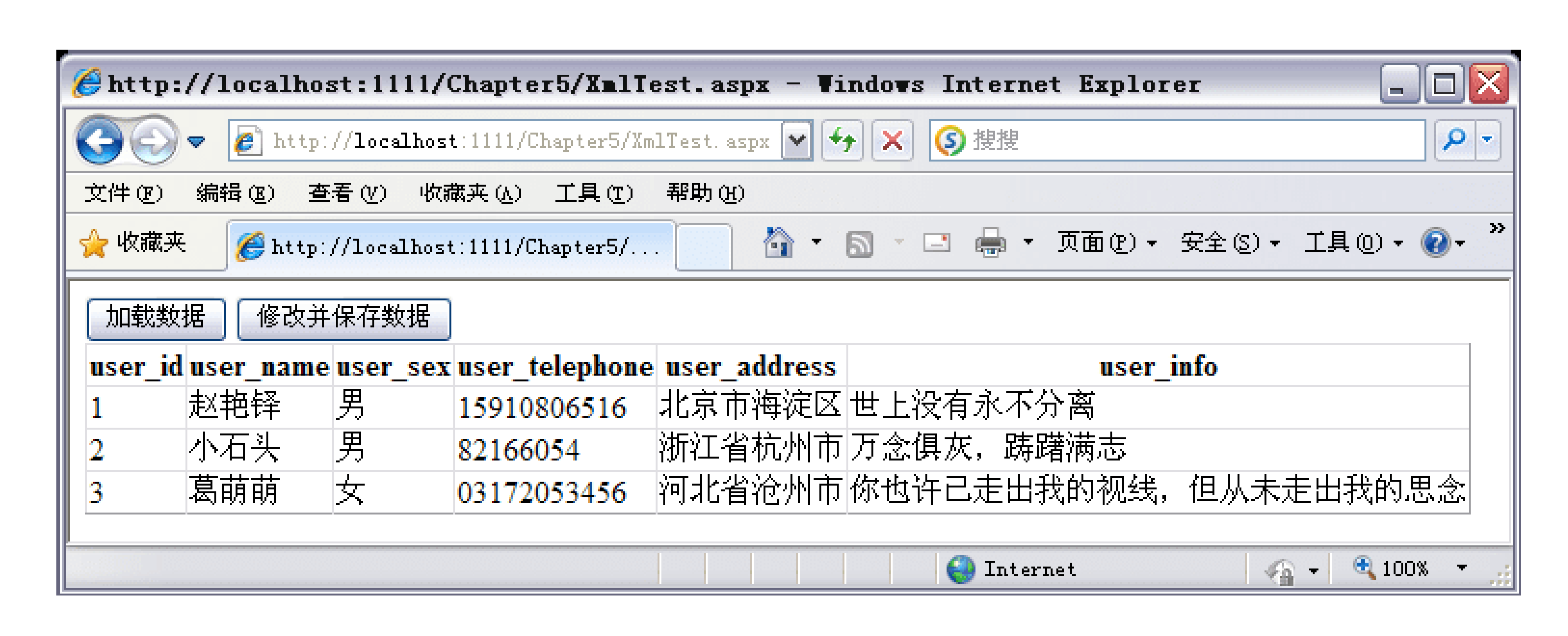


图 5-47　加载 XML 文件中的数据

(6) 单击“修改并保存数据”按钮，修改 user_id 为奇数的用户的 user_info 值并保存，然后重新加载并显示，如图 5-48 所示。

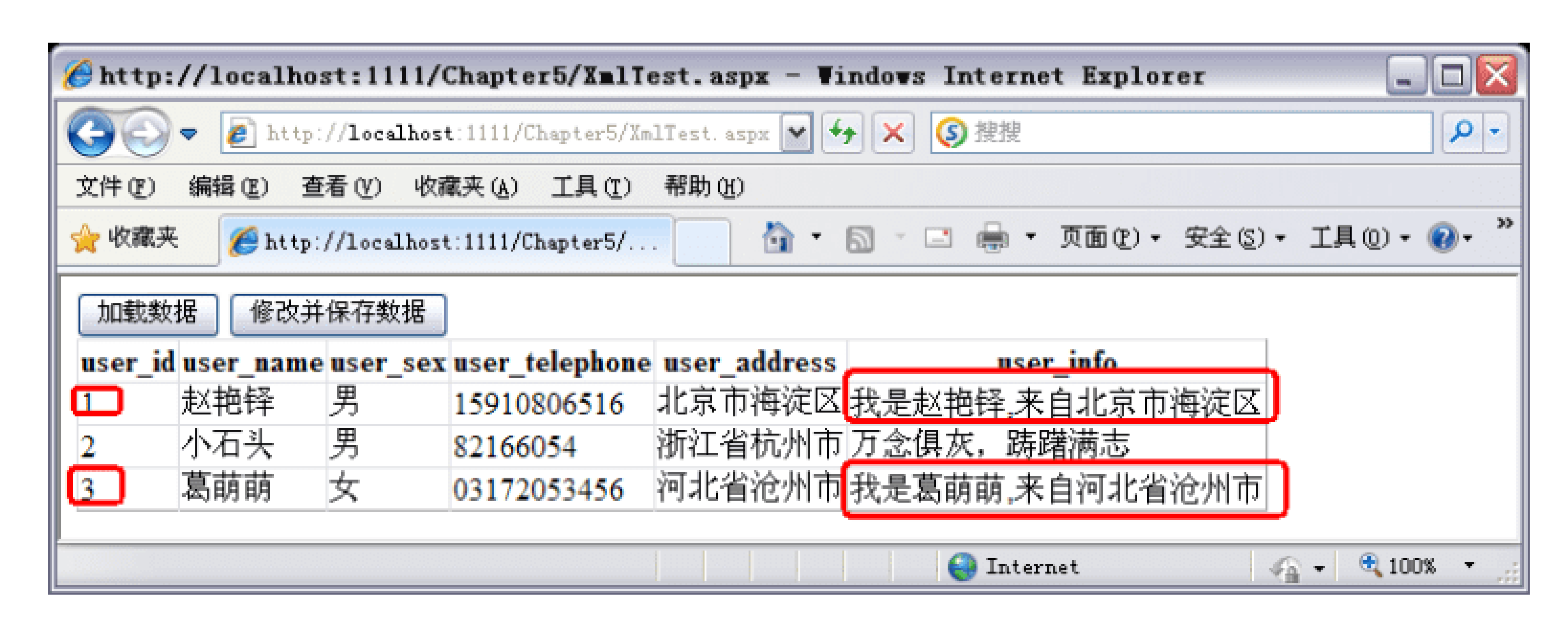


图 5-48　修改并保存数据

5.5.3　将数据库中的数据转换成 XML 文档

使用 XML 做数据交换可以使应用程序更具有弹性，因为可以用位置(与普通文件一样)或用元素(从数据库)来存取 XML 数据。

如果数据集 DataSet 中填充的是数据库中的数据，那么调用 WriteXml 方法后，即可将数据库数据转换成 XML 文档。

例 5-12：查询数据库中 Z_MESSAGE 表中的数据，并将其转换成 XML 文档。

(1) 启动 VWD 2010，打开网站 Chapter5。

(2) 通过“添加新项”命令，添加一个名为 Db2Xml.aspx 的页面，切换到页面的“设计”视图，添加一个 Button 控件、一个 Label 控件和一个 GridView 控件，设置 Button 控件的 Text 属性为“保存数据到 XML”。

(3) 切换到后台代码文件，添加对 ADO.NET 命空间的引用，代码如下：

```
using System.Data.SqlClient;
using System.Data;
using System.Web.Configuration;
```

(4) 为按钮控件添加单击事件处理程序，使用 SqlDataAdapter 填充数据集，将学生表 Student 中的数据保存到 XML 文件中，并显示在 GridView 控件中，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    string strConn =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
    SqlConnection con = new SqlConnection(strConn);
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        SqlDataAdapter sqld = new SqlDataAdapter("SELECT * FROM Z_MESSAGE", con);
        sqld.Fill(ds, "msg");//用 Fill 方法填充 DataSet
        GridView1.DataSource = ds.Tables["msg"];
```

```
                GridView1.DataBind();
                ds.WriteXml(Server.MapPath("message.xml"));
                Label1.Text = "保存数据到 XML 文件成功！";
            }
            catch (Exception ex)
            {
                Label1.Text = "操作失败，失败原因：" + ex.Message;
            }
            finally
            {
                con.Close();
                con = null;
            }
        }
```

(5) 编译并运行程序，在浏览器中打开 Db2Xml.aspx 文件，单击“保存数据到 XML”按钮，读取数据表 Z_MESSAGE 中的内容，并显示在 GridView 控件中，如图 5-49 所示。

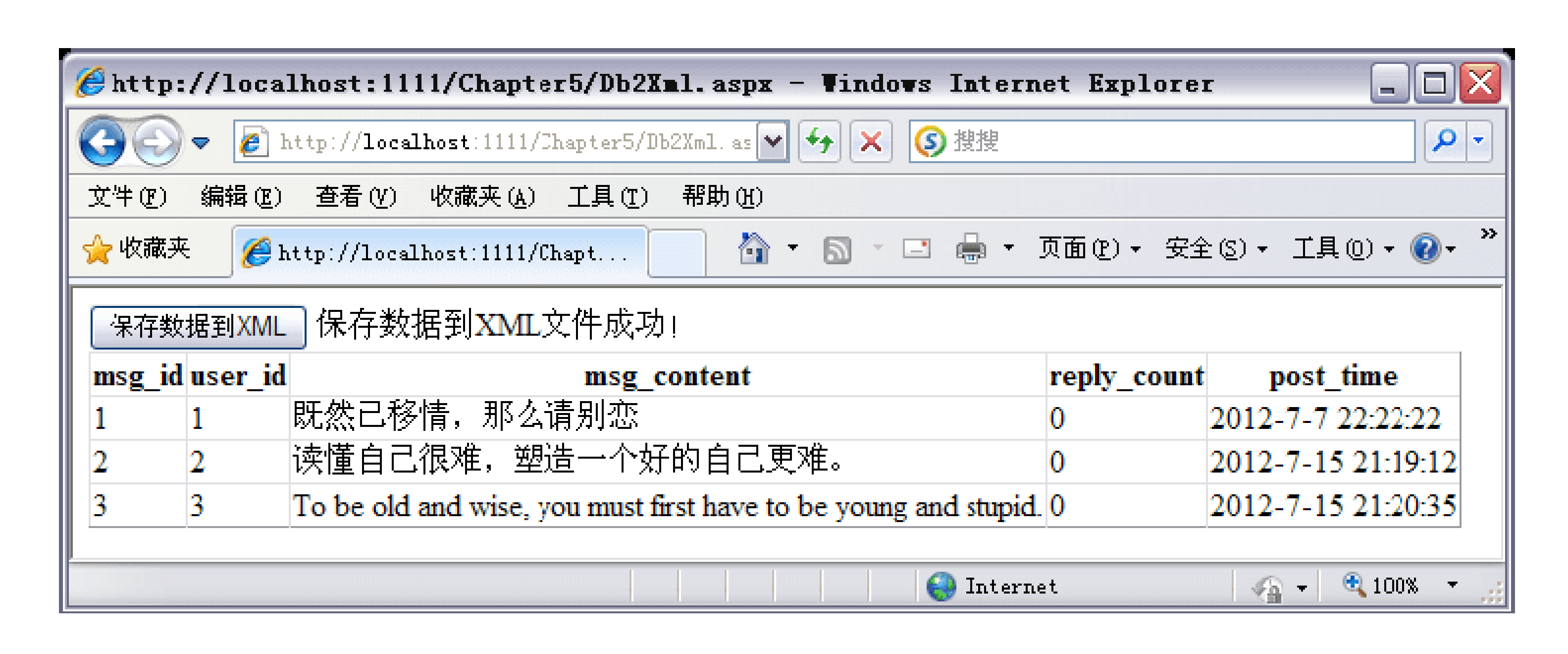


图 5-49　保存数据到 XML 文件中

(6) 在网站跟目录，打开程序运行时保存的 message.xml 文件，内容如下：

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <msg>
    <msg_id>1</msg_id>
    <user_id>1</user_id>
    <msg_content>既然已移情，那么请别恋</msg_content>
    <reply_count>0</reply_count>
    <post_time>2012-07-07T22:22:22+08:00</post_time>
  </msg>
  <msg>
    <msg_id>2</msg_id>
    <user_id>2</user_id>
    <msg_content>读懂自己很难，塑造一个好的自己更难。</msg_content>
    <reply_count>0</reply_count>
    <post_time>2012-07-15T21:19:12.787+08:00</post_time>
```

```
    </msg>
    <msg>
      <msg_id>3</msg_id>
      <user_id>3</user_id>
      <msg_content>To be old and wise, you must first have to be young and stupid.</msg_content>
      <reply_count>0</reply_count>
      <post_time>2012-07-15T21:20:35.083+08:00</post_time>
    </msg>
  </NewDataSet>
```

可以看出，这个文档保存了 Z_MESSAGE 数据表中所有的数据。其中使用<NewDataSet>作为根结点标记，<msg>作为每个记录的标记(msg 是 sqld.Fill(ds, "msg")语句中使用的名字)。另外，每个字段的名字作为数据元素的标记名。

5.6　本章小结

处理数据库的能力是对 Web 开发技能的一项很好的补充。ASP.NET 应用程序的数据访问是通过 ADO.NET 进行的，ADO.NET 可以使 Web 应用程序从各种数据源中快速访问数据。本章首先介绍了数据库的基本知识、使用 VWD 的内置数据库工具创建这些表以及使用 SQL 语言查询和操纵数据；接着介绍 ADO.NET 访问 SQL Server 数据库的方法；然后讨论了 VWD 提供的数据控件和数据绑定技术；最后介绍了 ADO.NET 访问 XML 数据。通过本章的学习读者应该了解基本的数据库操作，能够使用 SQL 从数据库中检索指定的数据，掌握如何在 ASPX 页面中访问和操作数据源，以及数据信息的显示与更新，学会设计主-从页面显示数据信息。

5.7　思考和练习

1. 什么是 SQL 语言，SQL 语言具有哪些特点和功能？
2. GROUP BY 子句用于对查询结果进行分组，如果分组后还要求按一定的条件对这些组进行筛选，最终只输出满足指定条件的组，需要使用什么指定筛选条件？
3. 如何使用 ADO.NET 调用存储过程。
4. ASP.NET 提供了哪些数据源控件？
5. 要使用 GridView 控件的排序功能，需要设置什么属性？
6. 对于简单的数据绑定，必须调用页面或者控件的什么方法才能使绑定生效？
7. DetailsView 控件和 FormView 控件有什么相同点，又有什么区别？
8. DataKeyNames 属性有什么作用？

第6章 LINQ

LINQ 是一种与.NET Framework 中使用的编程语言紧密集成的新查询语言。使用 LINQ 可以直接通过代码查询多种数据源中的数据。本章将主要介绍 LINQ 的语言及其语法，在 ASP.NET 项目中使用 LINQ 查询数据的许多方法，最后还接送了数据控件和 LINQ 结合使用的方法与技巧。通过本章的学习，读者将掌握 LINQ 的基本语法以及 LINQ to EF 的具体应用。

本章学习目标：

- LINQ 及其语法
- LINQ 的各种形式及其适用场合
- 了解 ADO.NET Entity Framework
- 使用 EntityDataSource 控件来访问 EF
- ListView 控件和 DataPage 控件的使用

6.1 LINQ 简介

LINQ，即语言集成查询(language-integrated query)，是一种与.NET Framework 中使用的编程语言紧密集成的新查询语言。它是在.NET 3.5 以后引入的，所以在.NET 2.0 及以前的版本中是不能使用 LINQ 的。

使用 LINQ 可以像使用 SQL 查询数据库的数据那样从.NET 编程语言中查询数据。事实上，LINQ 语法部分模仿了 SQL 语言，使熟悉 SQL 的编程人员更容易上手。

使用 LINQ 可以直接通过代码查询多种数据源中的数据。LINQ 之于.NET 应用程序编程就像 SQL 之于关系数据库。通过简单的、声明性的语法查询集合中匹配条件的对象。

LINQ 并不只是.NET Framework 的一个增件。相反，它被设计和实现为.NET 编程语言中的一部分。这意味着 LINQ 被真正集成到.NET 中，为查询数据提供了一个统一的方法，而不管数据的来源。另外，由于它被集成到语言中，而不是特定的项目类型中，所以可用于各种项目，包括 Web 应用程序、Windows Forms 应用程序和 Console 应用程序等。

LINQ 相关的类都放在 System.Linq 命名空间，所以要使用 LINQ，必须引入该命名空间：

```
using System.Linq;
```

由于 LINQ 非常强大，又极具潜力，因此它被集成到.NET Framewrok 的多个方面。下面将介绍不同的 LINQ 实现。

6.1.1 LINQ to Objects

这是语言集成的最基本形式。使用 LINQ to Objects 可以查询.NET Framework 中的几乎所有集合。实际上，使用 LINQ to Objects 对内存中的所以对象进行简单查询是非常方便的。

使用 LINQ 的查询通常由 3 个步骤组成：

(1) 获得数据源；

(2) 创建查询；

(3) 执行查询。

例 6-1：使用 LINQ 查询编号为奇数的用户信息，本例用到了泛型中的 Dictionary<K,V> 定义一组集合。

(1) 启动 VWD 2010，新建空网站 Chapter6。

(2) 通过“添加新项”对话框添加一个名为 LinqTest.aspx 的页面。

(3) 在 LinqTest.aspx 页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Dictionary<int, string> user = new Dictionary<int, string>();
    user.Add(1, "赵艳铎");
    user.Add(2, "小石头");
    user.Add(3, "葛萌萌");
    user.Add(4, "金百合");
    var result = from val in user
                 where val.Key % 2 == 1
                 select val;
    Response.Write("全部信息如下：<br/>");
    foreach (var item in user)
    {
        Response.Write("编号：" + item.Key + ";        姓名：" + item.Value);
        Response.Write("<br/>");
    }
    Response.Write("查询结果如下：<br/>");
    foreach (var item in result)
    {
        Response.Write("编号：" + item.Key + ";        姓名：" + item.Value);
        Response.Write("<br/>");
    }
}
```

上述代码是查询编号为奇数的用户信息并输出。

(4) 编译并运行程序，在浏览器中打开 LinqTest.aspx 页面，如图 6-1 所示。

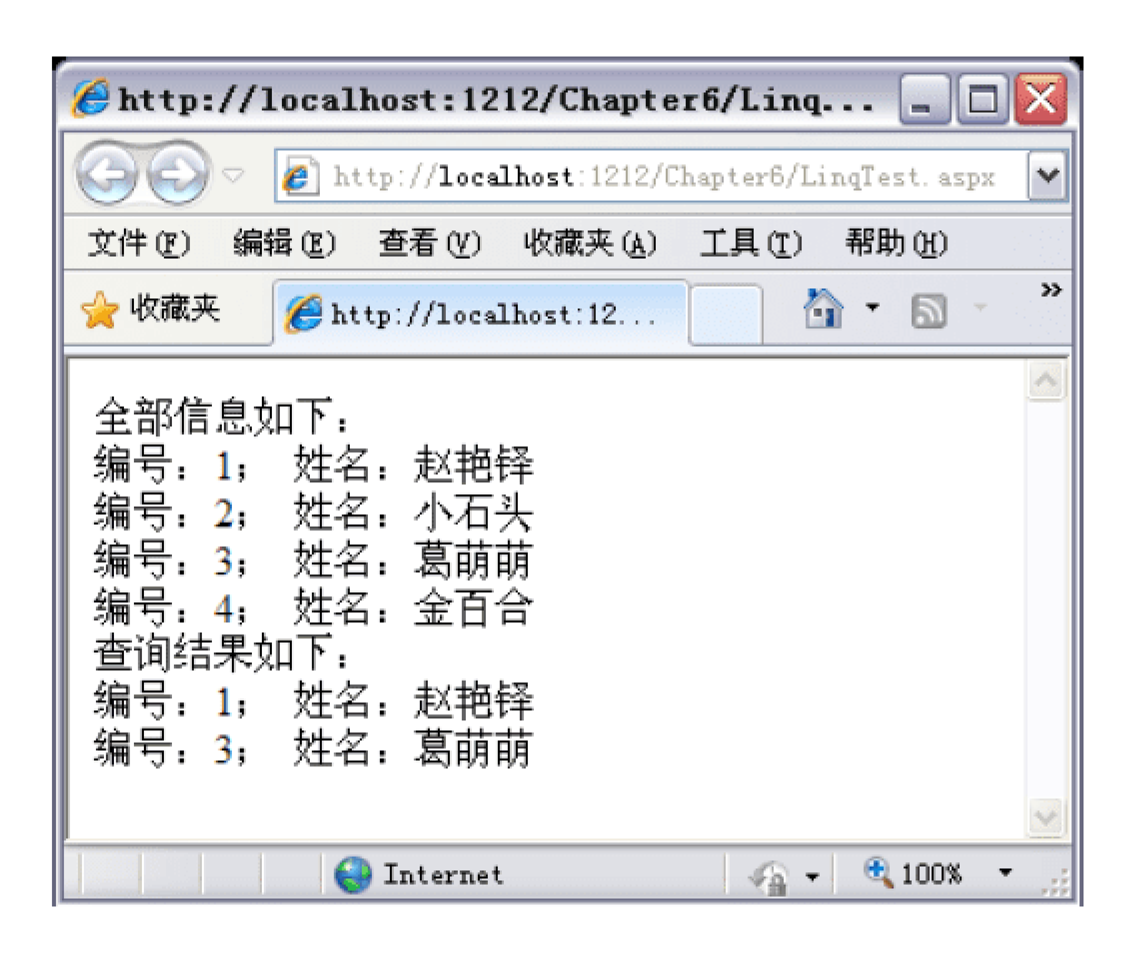


图 6-1 页面运行效果

6.1.2 LINQ 与泛型

LINQ 查询是建立在泛型这种数据类型的基础之上的，泛型是从.NET Framework 2.0 开始引入的。虽然编程人员无须深入了解泛型技术就可以开始编写 LINQ 查询，但是了解下面的两个泛型的基本概念有助于帮助读者理解其工作原理。

(1) 当创建泛型集合类(如 List<T>)的实例时，需要将 T 替换为集合中指定的对象类型，如字符串集合表示为 List<string>。因为泛型集合是强类型的，所以比将元素存储为 Object 类型的集合要强大得多。如果试图将一个 int 类型的对象添加到 List<string>，则会产生编译错误。

(2) IEnumerable<T>表示的是一个接口，通过该接口可以使用 foreach 语句来遍历泛型集合类。LINQ 查询变量可以类型化为 IEnumerable<T>或者它的派生类，如 IQueryable<T>。

为了避免使用泛型语法，可以使用匿名类型来声明查询，即使用 var 关键字来声明查询。var 关键字指示编译器通过查看在 from 子句中指定的数据来推断查询变量的类型。如在例 6-1 中声明的 result 和 item 变量。

在例 6-1 中定义的泛型类是 Dictionary<K，V>。该类型用于定义键/值(Key/Value)对的集合。与 List<T>不同，这个类需要实例化两个类型，分别用于键和值，以表示集合中的各项。

实例化 Dictionary<K, V>对象后，就可以对它执行与继承自 DictionaryBase 的类相同的操作，例如，可以使用强类型化的 Add()方法添加键/值对；可以使用 Keys 和 Values 属性迭代集合中的键和值；也可以迭代集合中的各个项，把每个项作为一个 KeyValuePair<K, V> 实例来获取。

6.1.3 LINQ to XML

LINQ to XML 是读、写 XML 的一种新的.NET 方法。现在，可以在应用程序中编写直接针对 XML 的 LINQ 查询，而不是使用普通的 XML 查询语言，如 XSLT 或 XPath。

在 System.Xml.Linq 命名空间中定义了很多 LINQ to XML 的类，这些类的结构关系如图 6-2 所示。

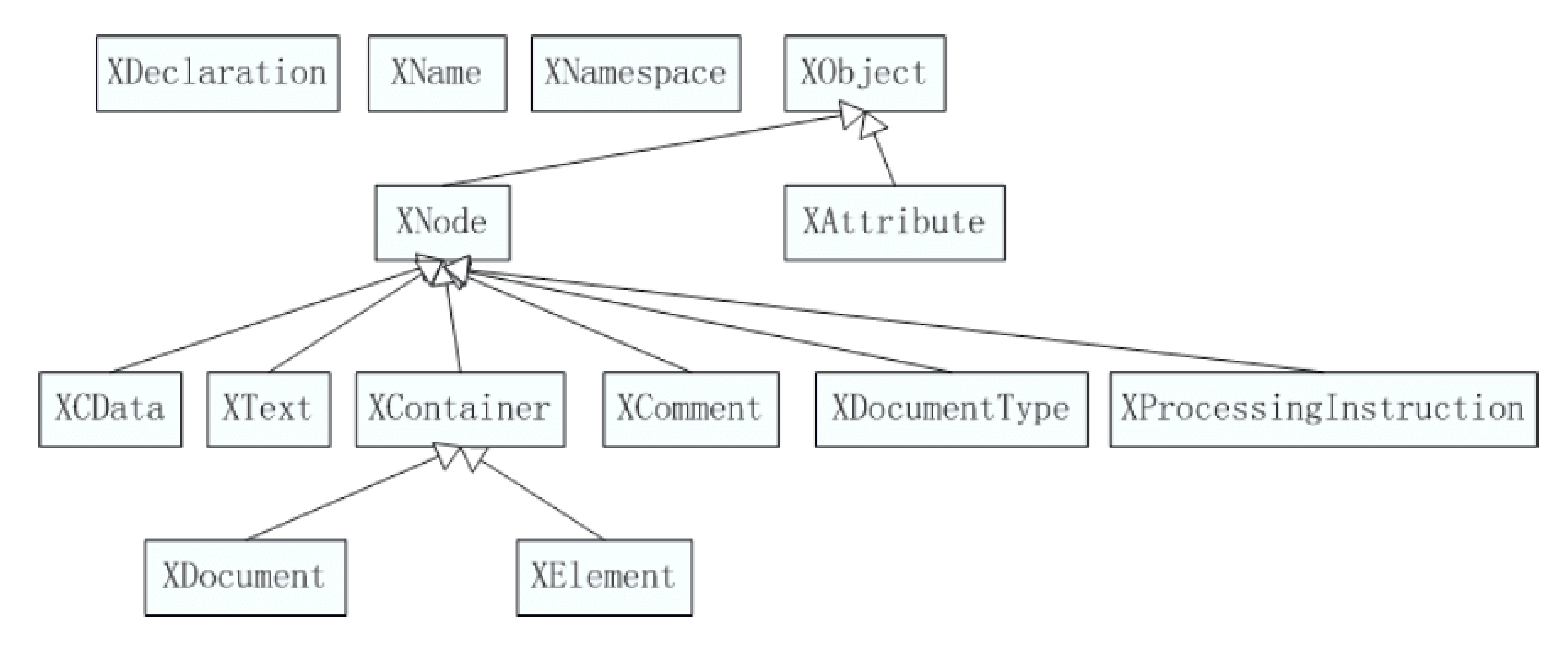


图 6-2　LINQ to XML 的类结构关系

其中 XElement 类是 LINQ to XML 中最基础的类，使用它可以创建一个 XMl 元素，使用 XAttribute 类可以为元素添加属性，使用 XNamespace 类可以为 XML 定义命名空间。

例 6-2：使用 LINQ to XML 读取 XML 文件中的数据，并查询符合条件的信息将其保存到结果 xml 中。

(1) 启动 VWD 2010，打开网站 Chapter6。

(2) 通过“添加新项”对话框添加一个名为 Users.xml 的 XML 文件，其内容与第 5 章中的 Users.xml 一样。

(3) 继续添加一个名为 Linq2Xml.aspx 的页面，并在页面中添加一个 Label 控件，用于显示读取到的 XML 数据。

(4) 切换到页面的后台代码文件，引入命名空间 System.Xml.Linq，然后在页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "<table border=\"1\" cellpadding=\"1\"><tr align=\"center\">";
    Label1.Text += "<td>用户 ID</td><td>姓名</td><td>性别</td><td>电话</td>";
    Label1.Text += "<td>所在地</td><td>个人介绍</td></tr>";
    XElement users = XElement.Load(Server.MapPath("Users.xml"));
    foreach (XElement user in users.Elements())
    {
        Label1.Text+="<tr>";
        foreach(XNode node in user.Nodes())
            Label1.Text += "<td>" + node + "</td>";
        Label1.Text += "</tr>";
    }
    Label1.Text += "</table>";
    //查询性别为“男”的用户节点
    var result = from user in users.Nodes()
                 where ((XElement)user).Element("user_sex").Value == "男"
```

```
                    select user;
            Label1.Text += "<br/>性别为“男”的用户信息如下：<br/><table border=\"1\"";
            Label1.Text += " cellpadding=\"1\"><tr align=\"center\"><td>用户 ID</td><td>姓名
</td>";
            Label1.Text += "<td>性别</td><td>电话</td><td>所在地</td><td>个人介绍</td></tr>";
            foreach (XElement user in result)
            {
                Label1.Text += "<tr>";
                foreach (XNode node in user.Nodes())
                    Label1.Text += "<td>" + node + "</td>";
                Label1.Text += "</tr>";
            }
            Label1.Text += "</table>";
        }
```

上述代码中首先加载 XML 文件中的数据，并以表格形式显示，然后查询其中性别为“男”的用户节点并显示在下方的表格中。

(5) 编译并运行程序，在浏览器中打开 Linq2Xml.aspx 页面，效果如图 6-3 所示。

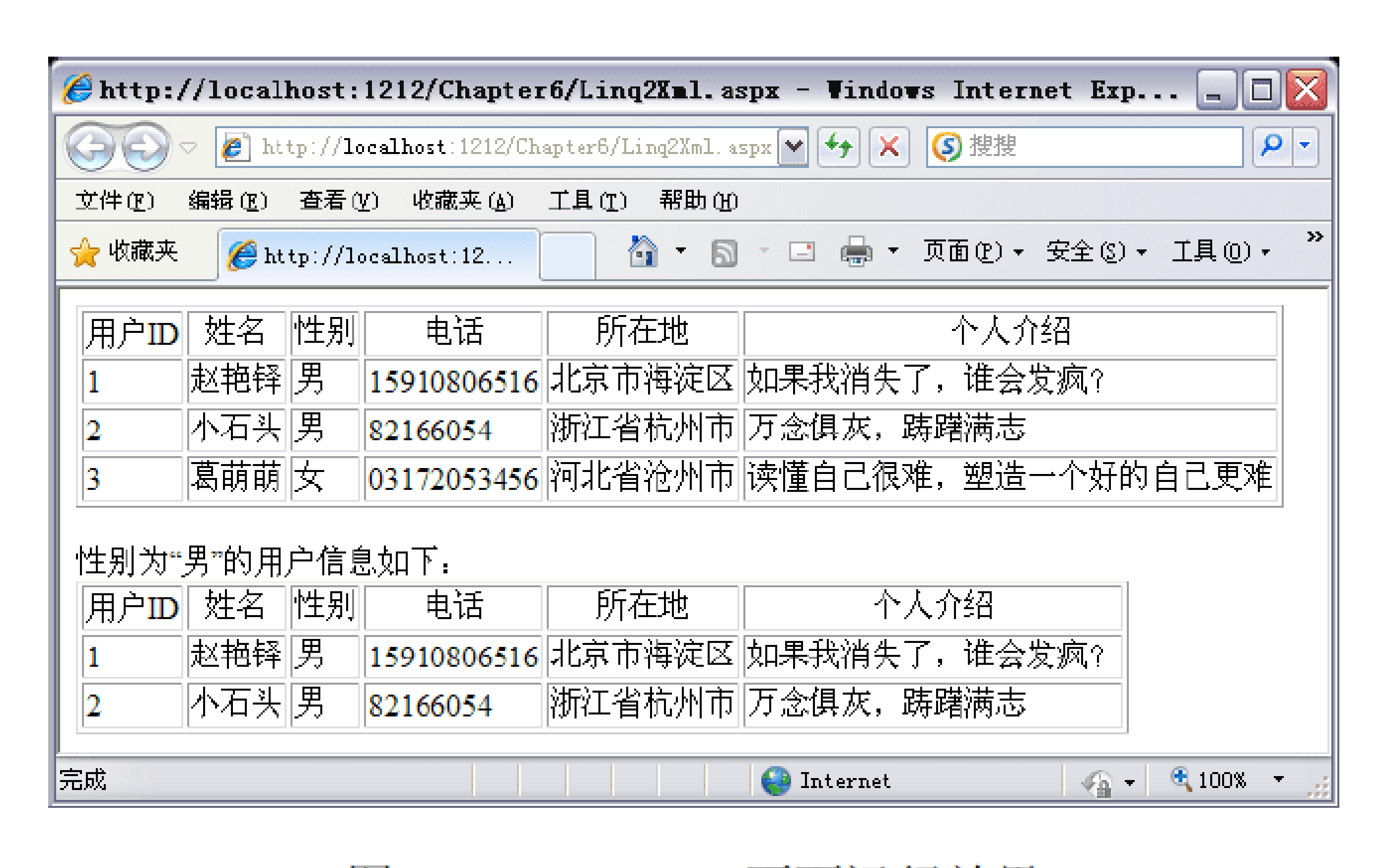


图 6-3　Linq2Xml 页面运行效果

6.1.4　LINQ to ADO.NET

ADO.NET 是.NET Framework 的一部分，它允许访问数据、数据服务(像 SQL Server)和其他许多不同的数据源。使用 LINQ to ADO.NET，可以查询与数据库相关的信息集，包括 LINQ to Entities、LINQ to DataSet 和 LINQ to SQL。

- LINQ to Entities 是 LINQ to SQL 的超集，比 LINQ to SQL 有更丰富的功能。不过，对于大多不同类型的应用程序来说，LINQ to SQL 已经足够了。
- LINQ to DataSet 允许对 DataSet 编写查询。
- LINQ to SQL 允许在.NET 项目中编写针对 Microsoft SQL Server 数据库的面向对象的查询。LINQ to SQL 将 LINQ 查询转换为 SQL 语句，然后再发送到数据库中执行 CRUD 的 4 种操作。在 ASP.NET 4 中，Microsoft 已经表示不会再积极开发 LINQ to SQL，这是因为 LINQ to SQL 与 Entity Framework(EF)在功能上有很大的重叠。在

LINQ to SQL 中能实现的操作在 EF 中也能实现。但是，与 LINQ to SQL 相比，EF 的功能要强大得多，并且功能要丰富得多。正因为如此，Entity Framework 是比 LINQ to SQL 更好的选择。本章将重点讨论 LINQ to Entity Framework。

6.2　ADO.NET Entity Framework(EF)

通过使用 ADO.NET Entity Framework(EF)，可以把许多数据库对象(如表)转换成可以在代码中直接访问的.NET 对象。然后就可以在查询中或者直接在数据绑定中使用这些对象。EF 也允许执行相反操作，首先设计一个对象模型，然后让 EF 创建必要的数据库结构。

6.2.1　ADO.NET EF 简介

通过 ADO.NET EF，在使用数据库时不必编写大量代码。ADO.NET Entity Framework 在后台大量使用 LINQ。使用 EF 十分简单，并且十分灵活。通过使用关系图设计器，可以将表等数据库对象拖放到实体模型中。放到关系图中的对象将成为可用的对象。例如，将 Z_USER 表放到关系图中就将得到一个强类型的 Z_USER 类。可以使用 LINQ 查询或者其他方式创建这个类的实例。

当把多个相关的数据库表放到关系图中时，设计人员可以观察到表之间的关系(如表 Z_USER 和 Z_USER_FUN)，然后在对象模型中复制这些关系。例如，使用某些 LINQ to EF 查询在代码中创建了一个 Z_USER_FUN 实例，那么就可以访问它的 Z_USER1 属性，进而可以访问 user_login 等属性：

```
Label1.Text = zUserFun.Z_USER1.user_login;
```

类似地，也可以访问某个用户的所有听众集合 Z_USER_FUN1s，以便将其绑定到数据绑定控件：

```
Repeater1.DataSource = zUser.Z_USER_FUN1s;
```

现在还不需要深究这些语法，本章将详细介绍它们。

6.2.2　将数据模型映射到对象模型

通过使用 EF，可以把数据库项(如表、列和数据库中的关系)映射到应用程序的对象模型中的对象和属性。VWD 提供了“ADO.NET 实体数据模型”模板和很多相关的工具来尽可能简化这种映射。下面通过一个具体的示例来介绍如何通过 EF 把数据模型映射到对象模型。

例 6-3：创建 ADO.NET 实体数据模型，通过 LINQ 查询来访问底层表中的数据。

(1) 启动 VWD 2010，打开网站 Chapter6。

(2) 在“解决方案资源管理器”窗口中右击项目，从弹出的快捷菜单中选择“添加新项”命令，然后选择“ADO.NET 实体数据模型”模板，默认名称为 Model.edmx，单击“添加”按钮将其添加到项目中。此时会弹出如图 6-4 所示的提示框，提示该类文件通常放在 App_Code 文件夹中，单击“是”按钮将其放在 App_Code 文件夹中。也可以直接在 App_Code 文件夹上单击鼠标右击添加。



图 6-4　提示实体数据模型文件放在特殊文件夹中

(3) 接着，启动“实体数据模型向导”，第一步是“选择模型内容”，如图 6-5 所示。选择“从数据库生成”选项，然后单击“下一步”按钮。



图 6-5　“选择模型内容”对话框

(4) 在“选择您的数据连接”对话框中，从下拉列表框中选择第 5 章中使用的 WeiBo 数据库连接，并选中“将 Web.Config 中的实体连接设置另存为”复选框，如图 6-6 所示。

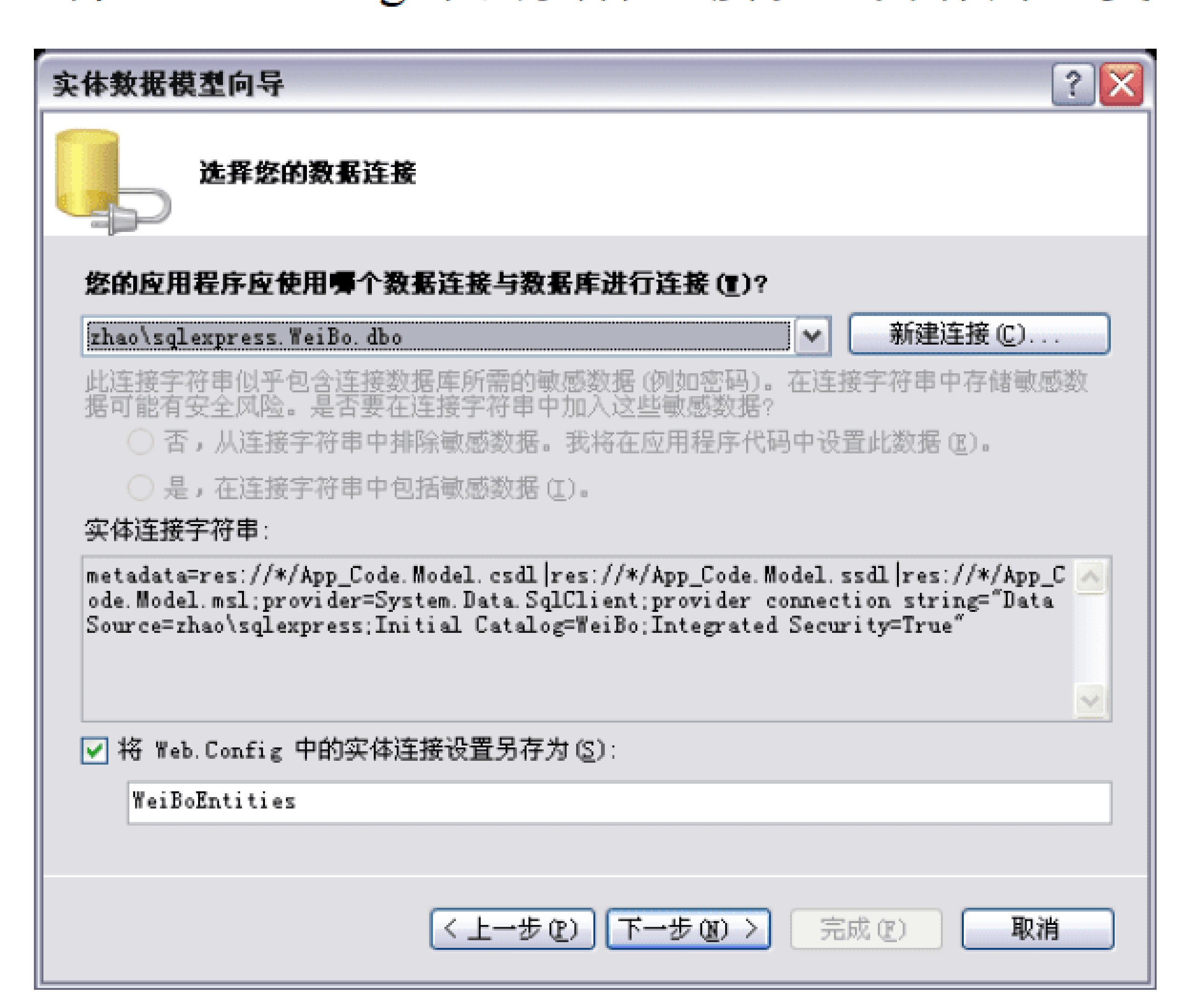


图 6-6　“选择您的数据连接”对话框

(5) 单击“下一步”按钮，进入“选择数据库对象”对话框。在该对话框中，选择表 Z_USER 和 Z_USER_FUN。该对话框下面有两个复选框，第一个表示在模型中自动将所有名称转换为单数或者复数形式，第二个是在模型中加入外键列，选中这两个复选框可以确保生成的对象模型中保留表之间的关系，如图 6-7 所示。如果没有选中模型中自动将名称转换为单数或者复数的复选框，则需要在生成模型以后手动进行修改。

提示：

此处的“模型命名空间”需要记住，稍后编写代码时要引用该命名空间。

(6) 单击“完成”按钮，即可将模型添加到站点中。VWD 将添加一个 Model.edmx 文件和 Model.designer.cs 后台代码文件，然后在主编辑器窗口中打开“实体设计器”，如图 6-8 所示。

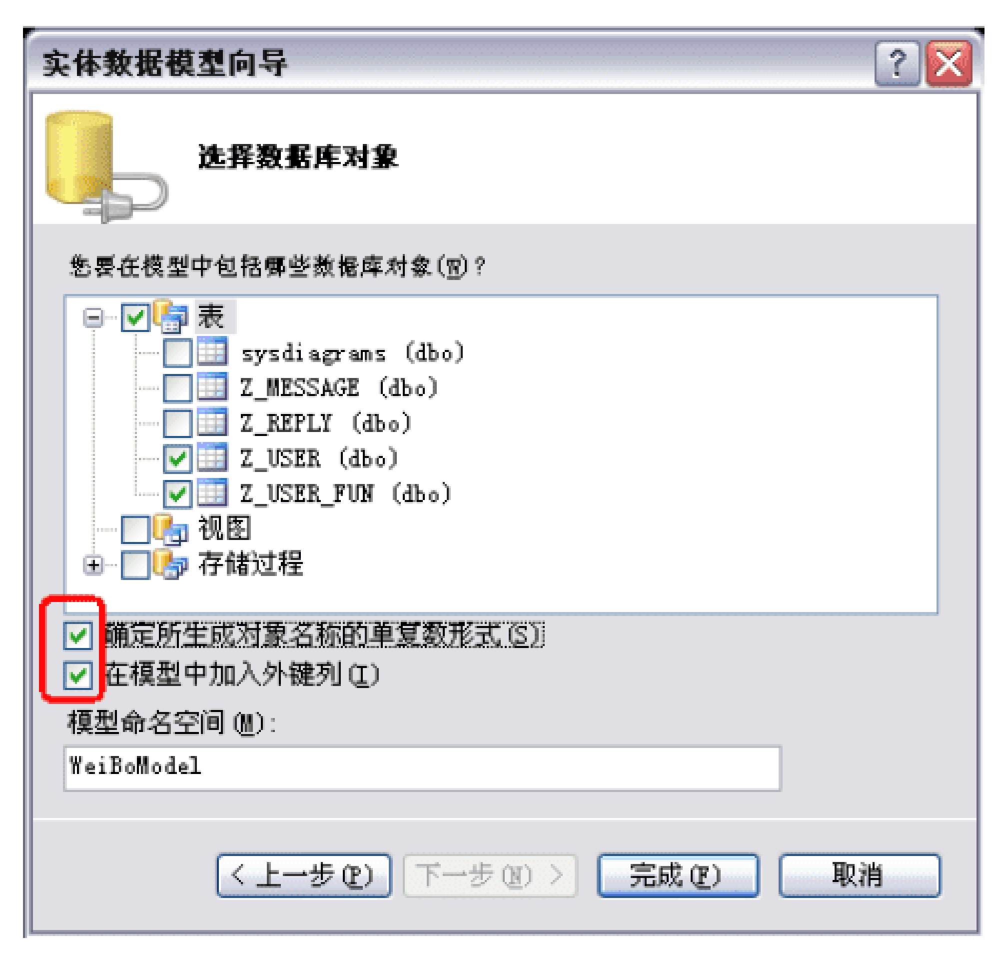


图 6-7　“选择数据库对象”对话框

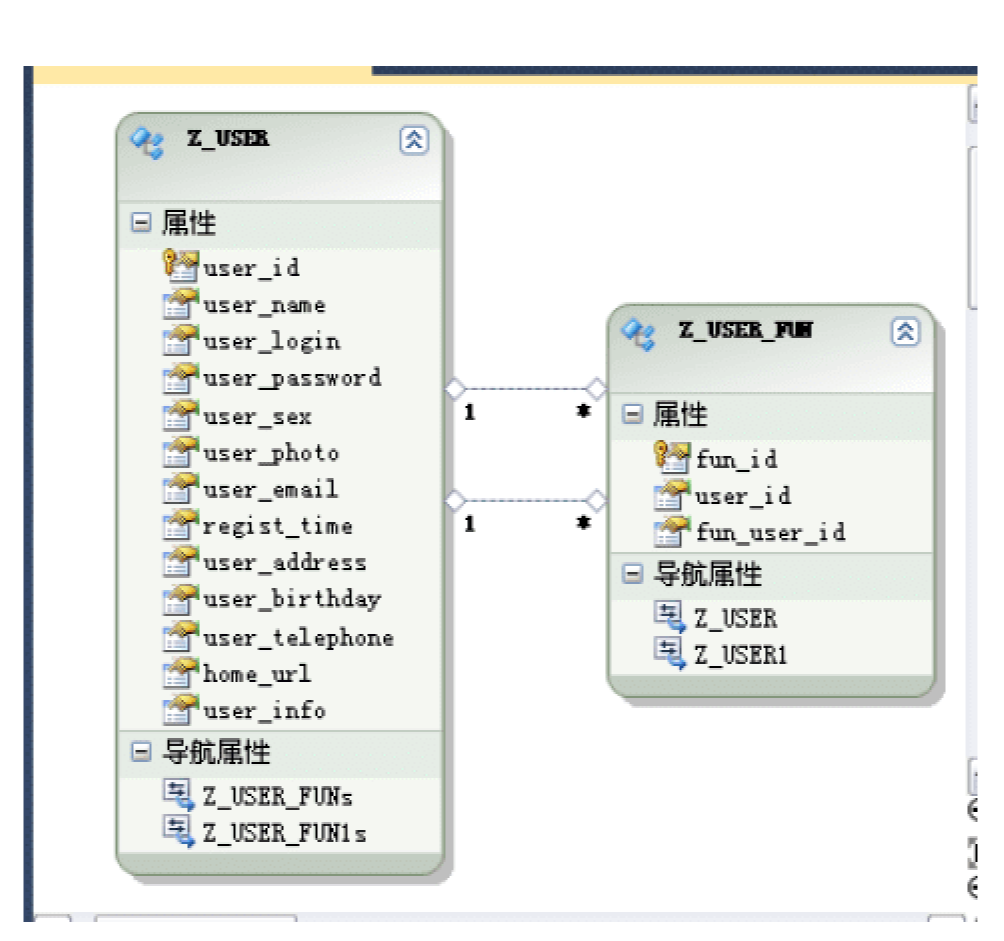


图 6-8　生成的实体对象模型关系图

这个关系图显示了基于数据库表生成的类，两个类之间的线表示了底层表之间的关系。二者是两个 1 对多的关系，即一个用户可以收听多个其他用户，同时也可以被多个用户收听。所以图中 Z_USER 类的导航属性是 Z_USER_FUNs 和 Z_USER_FUN1s (复数)，Z_USER_FUN 的导航属性是 Z_USER 和 Z_USER1(单数)。

技巧：

如果前面没有选中自动将名称转换为单数或者复数的复选框，那么在此可以分别选中生成的类，然后在“属性”面板中将“实体集名称”改为复数形式；再在类图中将导航属性重命名为复数形式。导航属性的修改需要根据具体的关系，对于一对多的关系中的“1”方不用修改为复数，而“实体集名称”则是所有类都要修改。

(7) 通过“添加新项”对话框添加一个名为 Linq2EF.aspx 的页面，并在页面中添加一个 DropDownList 控件和两个 GridView 控件，其中，DropDownList 控件用于显示所有用户的登录名，两个 GridView 控件分别用于显示选定用户的听众和收听人。

(8) 设置 DropDownList 控件的 AutoPostBack 属性为 True。

(9) 切换到页面的后台代码文件，引入前面生成实体数据模型时的命名空间

WeiBoModel，然后在页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        using (WeiBoEntities weiboEntity = new WeiBoEntities())
        {
            var allUser = from user in weiboEntity.Z_USER
                          select user;
            DropDownList1.DataSource = allUser;
            DropDownList1.DataTextField = "user_login";
            DropDownList1.DataValueField = "user_id";
            DropDownList1.DataBind();
        }
    }
}
```

说明：

using 中包装的代码用于创建在用完后必须释放(从内存中清除)的变量。由于 weiboEntity 保存了到 SQL Server 数据库的连接，因此将使用它的代码包装到 using 块中，这样对象会在块的末尾销毁。

(10) 添加 DropDownList 控件的 SelectedIndexChanged 事件处理程序，代码如下：

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    int userId = Int32.Parse(DropDownList1.SelectedValue);
    using (WeiBoEntities weiboEntity = new WeiBoEntities())
    {
        var users = from userFun in weiboEntity.Z_USER_FUN
                    where userFun.user_id == userId
                    select userFun.Z_USER1;
        GridView1.DataSource = users;//显示用户收听的人
        GridView1.DataBind();
        var userFuns = from userFun in weiboEntity.Z_USER_FUN
                    where userFun.fun_user_id == userId
                    select userFun.Z_USER;
        GridView2.DataSource = userFuns;//显示用户的听众
        GridView2.DataBind();
    }
}
```

上述代码中，第 1 个 LIQN 查询是查询 Z_USER_FUN 表中 user_id 为 DropDownList 控件中所选用户 id 的记录，然后显示记录对应的导航 Z_USER1，该属性为 Z_USER 实例。

第 2 个查询则是使用 fun_user_id 属性关联 DropDownList 控件中所选用户的 id。

(11) 编译并运行程序，在默认浏览器中加载 Linq2EF.aspx 页面，在下拉列表中选择某个用户，下方的 GridView 控件中将分别显示该用户的收听用户和听众信息，如图 6-9 所示。

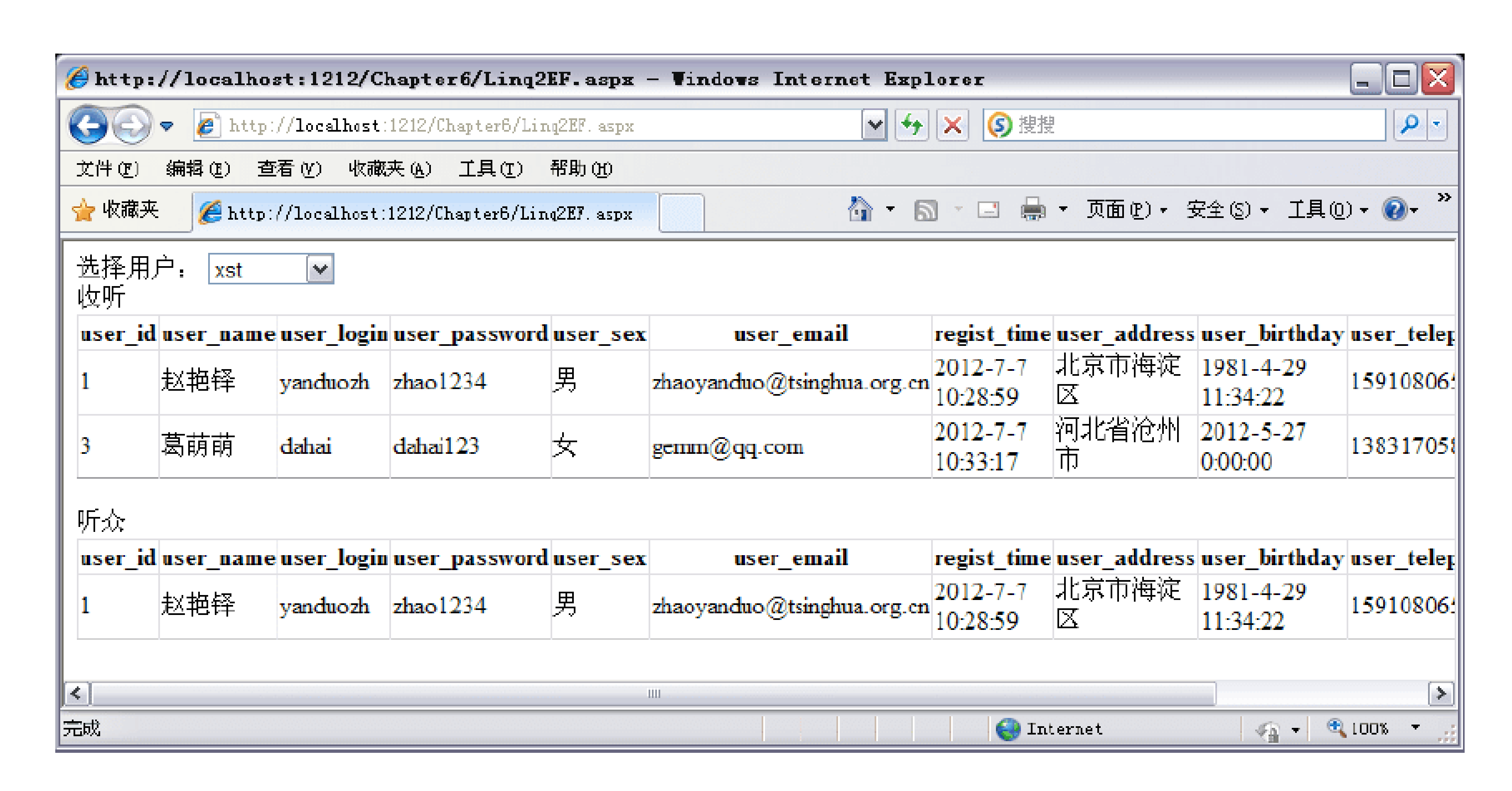


图 6-9　显示指定用户的收听和听众信息

通过这个例子可以看出，EF 提供了一个对象关系设计器(可以通过 VWD 访问)，允许基于数据库的表创建一个可通过代码访问的对象模型。只要将表拖至该设计器，VWD 就会创建可用于访问数据库中底层数据的代码，而无须自己编写大量代码。拖至设计器上的类存储在.edmx 文件和其后台代码文件中。其后台代码中包含了一个继承自 ObjectContext 的上下文类，而 ObjectContext 是 EF 中提供对数据库进行访问的主要实体。

在生成模型后，对其执行 LINQ 查询，即可从底层数据库中获取数据。要访问这些数据，需要一个 ObjectContext 类的实例，这要在代码的 using 块中创建。

6.3　LINQ 查询语法

前面的例子中使用了简单的 LINQ 查询，LINQ 的查询能力远高于此。本节将重点介绍 LINQ 查询语法。需要注意的是，LINQ 语法并不是专门为 Entity Framework 设计的。下面介绍的大多数 LINQ 概念同样适用于其他 LINQ 实现，如 LINQ to Objects 和 LINQ to ADO.NET 等。

6.3.1　基本语法

LINQ 支持大量的查询操作符——可用于选择、排序或筛选从查询返回的数据的关键字。尽管本章所有示例是在 LINQ to EF 的背景下讨论的，但也可以将它们应用到其他 LINQ 实现中。下面将使用例 6-3 中创建的名为 weiboEntity 的 ObjectContext 对象作为查询的数据源，系统地介绍 LINQ 查询的基本语法。

1. from

LINQ 查询表达式必须以 from 子句开头。尽管 from 子句不能算是标准查询操作符，因为它并不对数据进行操作而是指向数据，但它是 LINQ 查询中的一个重要元素，因为它定义了查询所执行的集合或数据源。在例 6-3 中的 from 子句指定了查询必须对 EF 中的 weiboEntity 对象所提供的 Z_USER 集合执行。

2. select

select 关键字用于从查询的源中检索对象。例如，下面的查询是选择已有类型的一个对象。

```
var allUser = from user in weiboEntity.Z_USER
              select user;
```

这一示例中的变量 user 是范围变量(range variable)，它只在当前查询中可用。通常在 from 子句中引入范围变量，然后在 where 和 select 子句中再次使用它来筛选数据，表明要选择的数据。尽管对于它可采用任意的名称，但通常看到的都是单个字母的变量，如 u，或所查询集合的单数形式 user。

3. where

和 SQL 中的 WHERE 子句一样，LINQ 中的 WHERE 子句允许筛选查询返回的对象。WHERE 子句的后面是一个逻辑表达式，例如，下面的查询将返回所有性别为“男”的用户信息：

```
var allUser = from user in weiboEntity.Z_USER
              where user.user_sex=="男"
              select user;
```

4. orderby

使用 orderby 可以对结果集合中的项进行排序。orderby 后面可以通过逗号分隔来指定多个条件。紧跟着的是可选的用来指定排序顺序的 ascending(升序)和 descending(降序)关键字，默认排序方式为升序。例如下列的查询将把结果按 user_login 升序，按 user_address 降序排列。

```
var allUser = from user in weiboEntity.Z_USER
              where user.user_sex=="男"
              orderby user.user_login ascending,user.user_address descending
              select user;
```

5. Sum、Min、Max、Average 和 Count

这些聚集运算符允许在结果集中的对象上进行数学计算。Sum 是求和运算符；Min 是求最小值运算符；Max 是求最大值运算符；Average 是求平均值运算符；Count 是计数运算符。例如，要检索指定用户的听众数，可以执行如下查询：

```
var userFuns = (from userFun in weiboEntity.Z_USER_FUN
                where userFun.fun_user_id == userId
                select userFun).Count();
```

注意：

Count 方法需要作用在整个结果集上，因此，需要将整个语句都扩到括号中，然后再调用 Count 方法。

6. Take、Skip、TakeWhile 和 SkipWhile

Take 和 Skip 允许在结果集中作子选择。这通常用于分页情况，其中只检索当前页面的记录。Take 从结果集中获取所请求数量的元素，然后忽略其余的；而 Skip 则相反，它跳过请求数量的元素，然后返回其余的。

在 EF 中，Take 和 Skip 操作符也被转换为 SQL 语句。这意味着分页是在数据库级发生的，而不是在 ASP.NET 页面中。这大大增强了查询的性能，特别是对于一些较大的结果集更是如此，因为不是所有的元素都必须从数据库转移到 ASP.NET 页面中。

注意：

要想使用 Skip，必须在跳过指定数量的记录之前，向查询中添加一个 orderby 子句来对结果进行排序。如果不显式地添加 orderby 子句，数据库就可能以无法预知的顺序返回结果，因此，在 LINQ 查询中添加 orderby 动作有助于从 Skip 方法获得一致的结果，因为在跳过和获取记录之前，会先对它们进行排序。

下面的例子显示了如何检索第二页的记录，假定页面大小为 10。

```
var allUser = (from user in weiboEntity.Z_USER
               orderby user.user_login ascending,user.user_address descending
               select user).Skip(10).Take(10);
```

和 Count 一样，该查询也被括在一对括号中，然后调用 Skip 和 Take 来获取请求的记录。

TakeWhile 和 SkipWhile 查询操作符的工作方式类似，但允许在特定条件满足时获取或跳过一些记录。遗憾的是，在 EF 中无法使用它们，但是通常可以通过给查询添加一个简单的 where 子句来解决这个问题。

7. Single 和 SingleOrDefault

Single 和 SingleOrDefault 操作符允许返回单个对象作为强类型化实例。如果已知查询只返回一条记录，将很有用，例如，通过 user_login 查询用户信息。

```
var oneUser = (from user in weiboEntity.Z_USER
               where user.user_login=="yanduozh"
               select user).Single();
```

如果请求的项未找到或是查询返回多个实例，Single 操作符就会引发异常。如果想让该方法返回 null(没找到)，或是返回相关数据类型的默认值(如 Integer 型的 0、Boolean 型的

False 等)，则使用 SingleOrDefault。

说明：

即使查询结果只有一条记录，如果未调用 Single，则仍会返回一个列表集合。通过使用 Single，可强制结果集为所查询类型的单个实例。

8. First、FirstOrDefault、Last 和 LastOrDefault

这些操作符允许返回特定对象序列对象中的第一个或最后一个元素。和 Single 方法一样，如果集合为空，First 和 Last 就会抛出异常，而 FirstOrDefault 和 LastOrDefault 则返回相关数据类型的默认值。

与 Single 不同的是，当查询返回多个项时，First、FirstOrDefault、Last 和 LastOrDefault 操作符并不抛出异常。

技巧：

EF 中并不支持 Last 和 LastOrDefault 查询。不过，通过使用 First 和降序排列可以实现与之相同的操作。

例如，下面的查询分别返回最新注册的用户和最早注册的用户信息：

```
var firstUser = (from user in weiboEntity.Z_USER
                 orderby user.regist_time ascending
                 select user).First();
var newUser = (from user in weiboEntity.Z_USER
                 orderby user.regist_time descending
                 select user).First();
```

6.3.2　使用匿名类型定形数据

到目前为止，在本章前几节中看到的查询都返回的是全类型。即查询返回了一个 Student 实例的列表(如 select 方法)或是一个数值(如 Count)。

不过，有的时候不需要这些对象的所有信息，或者想对某些信息进行映射或转换。例如，在例 6-3 中，希望在 GridView 的标题列显示中文描述信息。这时，可以通过匿名类型(anonymous type)来重新定义要显示的列。

匿名类型是一种不需要像使用其他类型(如类)时那样预先定义名称的类型。而是可以通过选择数据，然后让编译器推断其类型来进行构造。

创建匿名类型的方法很简单；不需要使用像 select user 这样的语句选择实际的对象，而是使用 new 关键字，然后在一对花括号中定义要选择的属性。例如：

```
var u = from user in weiboEntity.Students
        where user.user_login=="yanduozh"
        select new{user.user_name,user.user_info};
```

尽管这一类型是匿名的，不能通过名称直接访问，但编译器仍能推断其类型，对于在查询中选择的新属性提供了完全智能识别感知功能。

除了直接选择已有属性外，还可以创建属性值并提供不同的名称。例如修改例 6-3 中查询用户收听和听众信息的 LINQ 语句，创建一个新的匿名类型，用中文重命名所有列。修改后的代码如下：

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    int userId = Int32.Parse(DropDownList1.SelectedValue);
    using (WeiBoEntities weiboEntity = new WeiBoEntities())
    {
        var users = from userFun in weiboEntity.Z_USER_FUN
                    where userFun.user_id == userId
                    select new
                    {
                        用户 ID = userFun.Z_USER1.user_id,
                        姓名 = userFun.Z_USER1.user_name,
                        登录名 = userFun.Z_USER1.user_login,
                        性别 = userFun.Z_USER1.user_sex,
                        生日 = userFun.Z_USER1.user_birthday,
                        所在地 = userFun.Z_USER1.user_address,
                        Email = userFun.Z_USER1.user_email,
                        电话 = userFun.Z_USER1.user_telephone,
                        个人简介 = userFun.Z_USER1.user_info
                    };
        GridView1.DataSource = users;
        GridView1.DataBind();
        var userFuns = from userFun in weiboEntity.Z_USER_FUN
                       where userFun.fun_user_id == userId
                       select new
                       {
                           用户 ID = userFun.Z_USER.user_id,
                           姓名 = userFun.Z_USER.user_name,
                           登录名 = userFun.Z_USER.user_login,
                           性别 = userFun.Z_USER.user_sex,
                           生日 = userFun.Z_USER.user_birthday,
                           所在地 = userFun.Z_USER.user_address,
                           Email = userFun.Z_USER.user_email,
                           电话 = userFun.Z_USER.user_telephone,
                           个人简介 = userFun.Z_USER.user_info
                       };
        GridView2.DataSource = userFuns;
        GridView2.DataBind();
    }
}
```

页面的运行效果如图 6-10 所示。

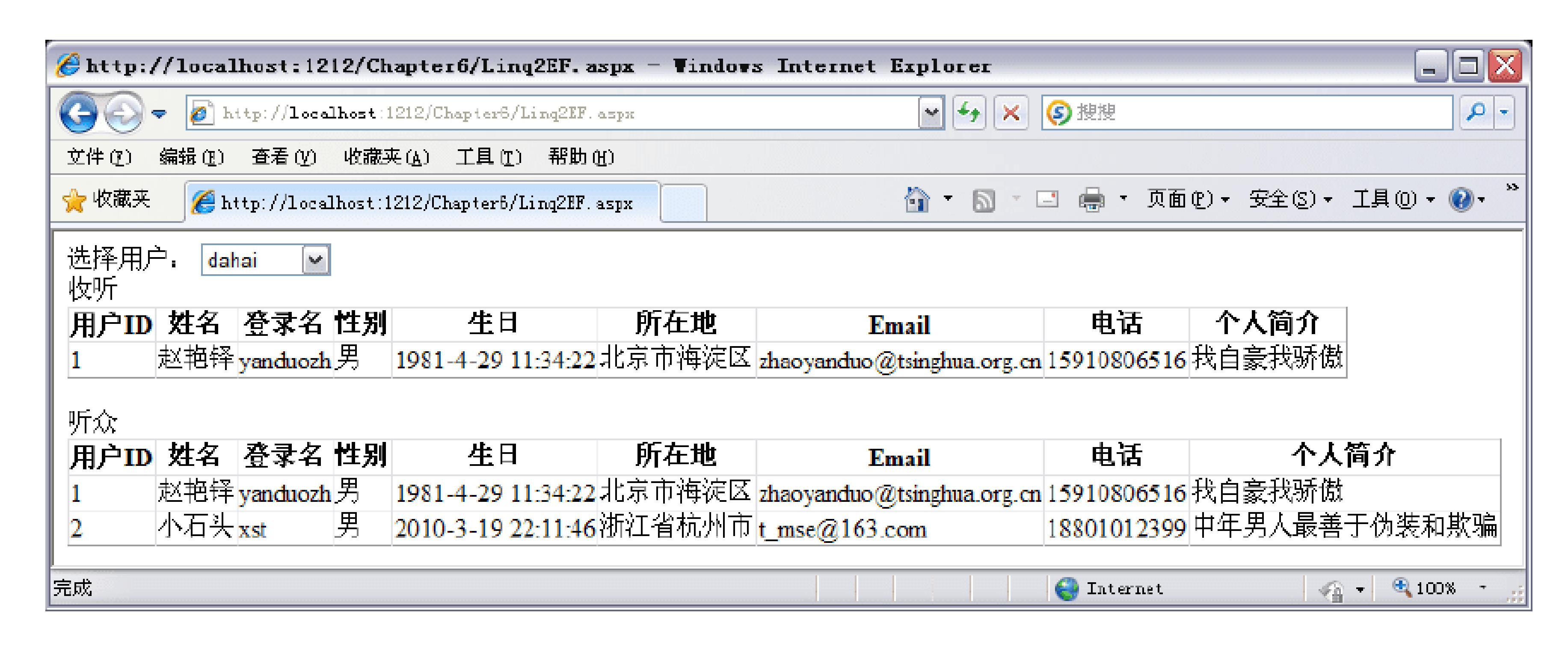


图 6-10　使用匿名类定形数据后的效果

6.4　使用数据控件和 LINQ

在前面的例子中，将 LINQ 查询的结果指派给控件的 DataSource 属性，然后调用 DataBind 方法即可显示在页面中。但是，这种方法只能显示数据，它不支持直接编辑、更新和删除数据。本节将介绍 EntityDataSource 控件和数据控件的绑定，使用这些控件可以很容易地实现编辑、更新和删除功能。

6.4.1　EntityDataSource 控件

EntityDataSource 和 SqlDataSource 以及其他数据源控件类似。EntityDataSource 控件之于 EF 就像 SqlDataSource 控件之于基于 SQL 的数据源，它提供了一个声明性的方法来访问支持 LINQ 的数据源模型。和 SqlDataSource 控件一样，EntityDataSource 提供了对 CRUD 操作的轻松访问，另外使数据排序和筛选也变得非常简单。

EntityDataSource 控件的主要属性如表 6-1 所示。

表 6-1　EntityDataSource 控件的主要属性

属　　性	描　　述
EnableDelete EnableInsert EnableUpdate	表明确定控件是否提供自动插入、更新和删除功能。如果启用，可以结合使用该控件和数据绑定控件(如 GridView 或 ListView)来支持数据管理
ContextTypeName	控件将使用的 ObjectContext 类的名称
EntitySetName	想使用的 EF 关系图中的表实体集名，如 Reviews
Select OrderBy Where	允许定义 EntityDataSource 控件对模型触发的查询。每个属性都映射到前面见到过的一个查询操作

和数据绑定控件一起，EntityDataSource 通过 EF 提供了对底层 SQL Server 数据库的完全访问。下面举例说明该控件的用法。

例 6-4：使用 EntityDataSource 作为数据控件的数据源。

(1) 启动 VWD 2010，打开网站 Chapter6。

(2) 通过“添加新项”对话框，添加名为 EntityDataSourceTest.aspx 的页面。

(3) 在 EntityDataSourceTest.aspx 页面的“设计”视图中添加一个 EntityDataSource 控件，在“EntityDataSource 任务”面板中选择“配置数据源”选项，开始配置数据源。

(4) 第一步是“配置 ObjectContext”，在此选择例 6-3 中创建的实体数据模型的命名连接 WeiBoEntities，如图 6-11 所示。

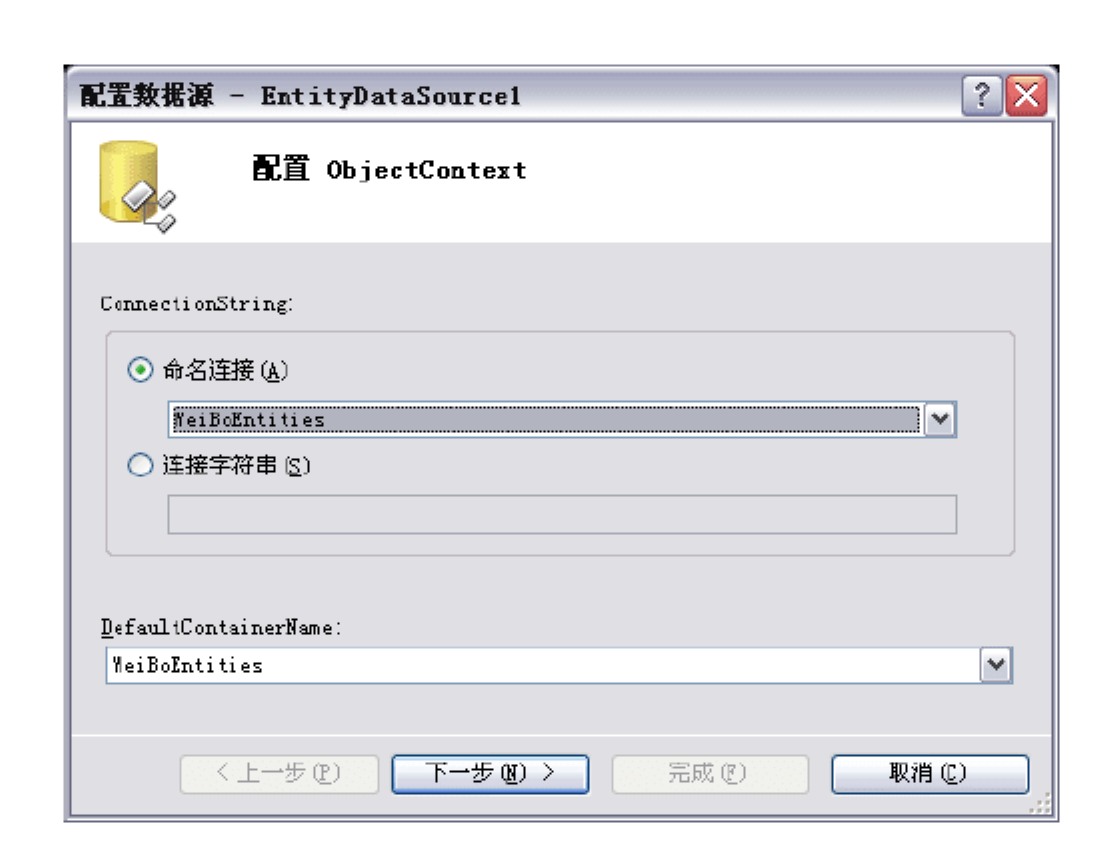


图 6-11　“配置 ObjectContext”对话框

(5) 单击“下一步”按钮，在“配置数据选择”对话框中选择 Z_USER 实体集，并选中下面的 3 个复选框，如图 6-12 所示。



图 6-12　“配置数据选择”对话框

(6) 单击“完成”按钮，完成数据源的配置。

(7) 在 EntityDataSourceTest.aspx 页面中添加一个 DetailsView 控件。设置控件的数据源为前面创建的数据源 EntityDataSource1，并选中“DetailsView 任务”面板中的“启用分页”、“启用插入”、“启用编辑”和“启用删除”复选框，然后通过“编辑字段”选项，编辑各列的 HeaderText 属性为中文描述。

(8) 通过“自动套用格式”选项为控件设置一个好看的样式。

(9) 无须编写任何代码，编译并运行程序，在浏览器中加载 EntityDataSourceTest.aspx 页面，效果如图 6-13 所示。

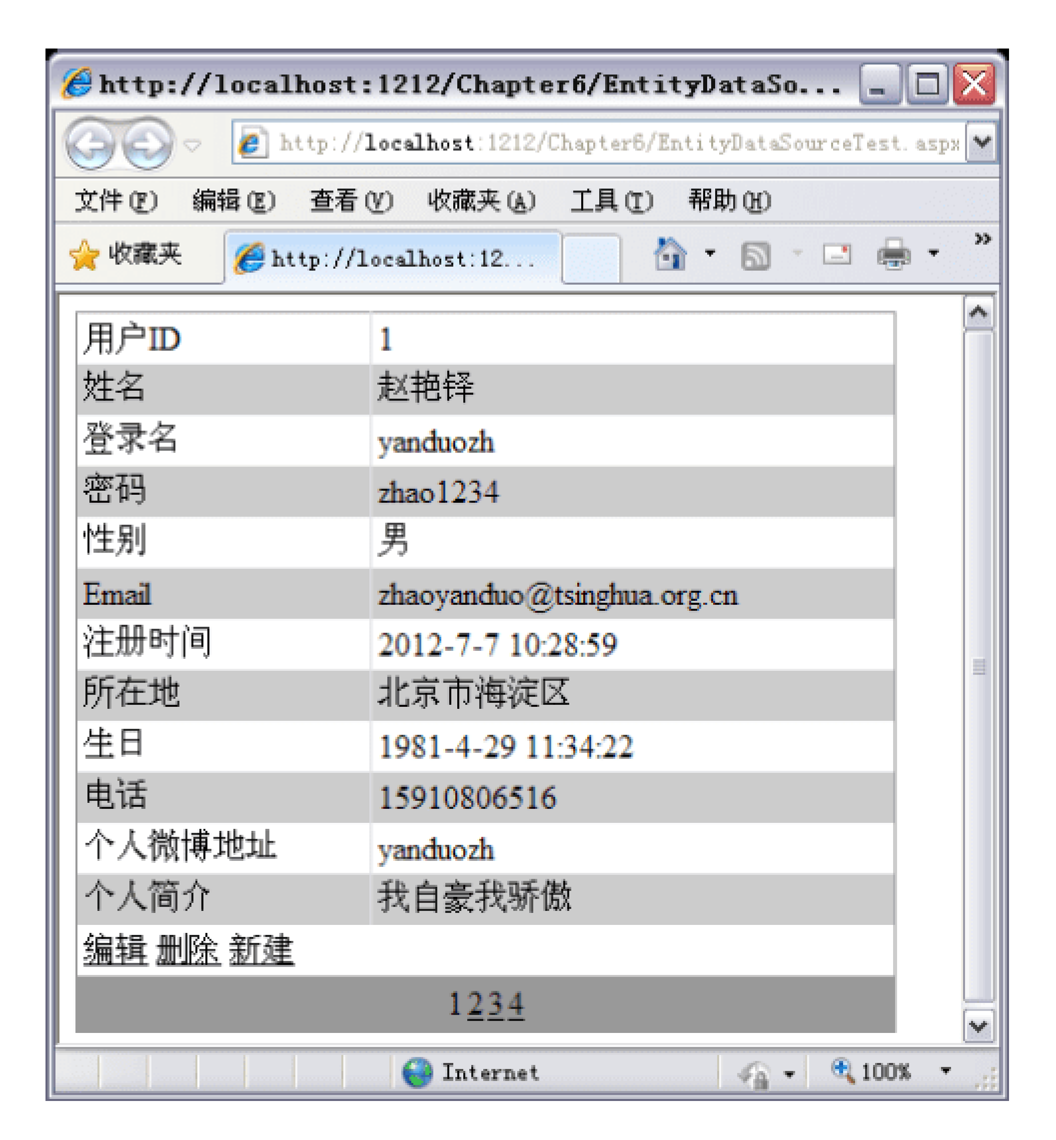


图 6-13　使用 EntityDataSource 控件页面运行效果

通过这个示例可以看出，创建和使用 EntityDataSource 数据源控件和使用 SqlDataSource 控件一样简单。

6.4.2　ListView 控件和 DataPager 控件

ListView 控件结合了 GridView 丰富的功能集和对 Repeater 提供的对标记的控制功能，并且还具有 DetailsView 的插入行为。ListView 可以以不同的格式显示数据，包括网格(像 GridView 那样的行和列)、项目符号列表和流格式等。

ListView 通过模板来显示和管理其数据。所有可用的模板如表 6-2 所示。

表 6-2　ListView 控件的可用模板

模　板	描　述
<LayoutTemplate>	作为控件的容器。它使得可定义一个放置单独数据项的位置。然后通过 ItemTemplate 和 AlternatingItemTemplate 表示的数据项作为该容器的子元素添加
<ItemTemplate> <AlternatingItemTemplate>	定义控件的只读模式。当一起使用时，它们可以创建一种“斑马纹效果”，其中奇偶行有着不同的外观(通常是不同的背景色)
<SelectedItemTemplate>	允许定义当前活动的或选择的项的外观
<InsertItemTemplate> <EditItemTemplate>	这两个模板允许定义用于插入和更新列表中的项的用户界面。通常，放置文本框、下拉列表和其他服务器控件等，并将它们与底层数据源绑定

(续表)

模　板	描　述
<ItemSeparatorTemplate>	定义放置在列表中项之间的标记，可用于在项之间添加线、图像或其他标记
<EmptyDataTemplate>	当控件中没有数据显示时显示该模板中的内容。可以是文本或其他标记
<GroupTemplate> <GroupSeparatorTemplate> <EmptyItemTemplate>	在高级表现场景中使用，其中数据可呈现在不同的组中

尽管这些模板看上去让人觉得需要编写大量代码才能使用 ListView，但事实并非如此。首先，VWD 2010 根据一些控件(如 EntityDataSource)提供的数据，创建了大部分的代码；其次，并不总是需要所有的模板。

除了模板之外，ListView 控件还有很多属性，如表 6-3 所示。

表 6-3　ListView 控件的主要属性

模　板	描　述
ItemPlaceholderID	放置在 LayoutTemplate 中的服务器端控件的 ID。当该属性引用的控件在屏幕上显示时，将由所有重复的数据项取代。它可以是一个真正的服务器控件，如<asp:PlaceHolder>；或者是一个简单的 HTML 元素，带有一个有效的 ID，其 runat 特性设置为 server(如<ul runat="server" id=" MainList" >)。如果不设置该属性，ASP.NET 会尝试找到 ID 为 itemPlaceholder 的控件并使用该控件
DataSourceID	页面上数据源控件的 ID，如 EntityDataSource 或 SqlDataSource 控件
InsertItemPosition	这一属性的枚举包括 3 个值(None、FirstItem 和 LastItem)，允许用于确定 InsertItem Template 的位置：在列表的开始或末尾，或者不可见

和其他数据绑定控件一样，ListView 也有大量事件。例如，在项插入到底层数据源前后触发的 ItemInserting 和 ItemInserted 事件。类似地，也有在更新和删除数据前后触发的事件。

ListView 控件的分页功能是通过 DataPager 控件实现的。DataPager 控件是 ASP.NET 3.5 引入的控件，可用它来扩展另一个数据绑定的控件。目前，只允许使用 DataPager 为 ListView 控件提供分页功能。

有两种方法将 DataPager 与 ListView 控件关联。

- 可以在 ListView 控件的<LayoutTemplate>中定义它，在这种情况下，DataPager 知道应自动为给哪个控件提供分页功能。
- 完全在 ListView 的外部定义它，在这种情况下，需要将 DataPager 的 PagedControlID 属性设置为有效 ListView 控件的 ID。

例 6-5：使用 ListView 控件和 DataPager 控件分页显示当前显示用户发表的所有信息。

(1) 启动 VWD 2010，打开网站 Chapter6。

(2) 通过“添加新项”对话框添加名为 Model2.edmx 的 ADO.NET 实体数据模型。实

体数据模型的命名连接为 WeiBoMsgEntities，选择的数据库对象是表 Z_USER 和 Z_MESSAGE，模型命名控件是 WeiBoMsgModel。

(3) 通过“添加新项”对话框添加名为 ListView.aspx 的页面。

(4) 在 ListView.aspx 页面的“设计”视图中添加两个 EntityDataSource 控件，使用 Model2.edmx 实体数据模型为这两个数据源控件配置数据源。

(5) 配置 EntityDataSource1 的数据源时，在“配置数据选择”对话框中选择 Z_USER 实体集，并选中下面的 3 个复选框。

(6) 配置 EntityDataSource2 的数据源时，在“配置数据选择”对话框中选择 Z_MESSAGE 实体集，并选中下面的 3 个复选框。

(7) 添加一个 FormView 控件，设置控件的数据源为 EntityDataSource1，并启用分页功能。

(8) 在“FormView 任务”面板中选择“编辑模板”选项，打开“编辑模板模式”，从下拉列表中选择“ItemTemplate”选项，将该模板中每一项提示文本都改为中文信息，并删除“密码”和“头像”两个字段(这两个字段将不显示在页面中)，如图 6-14 所示。

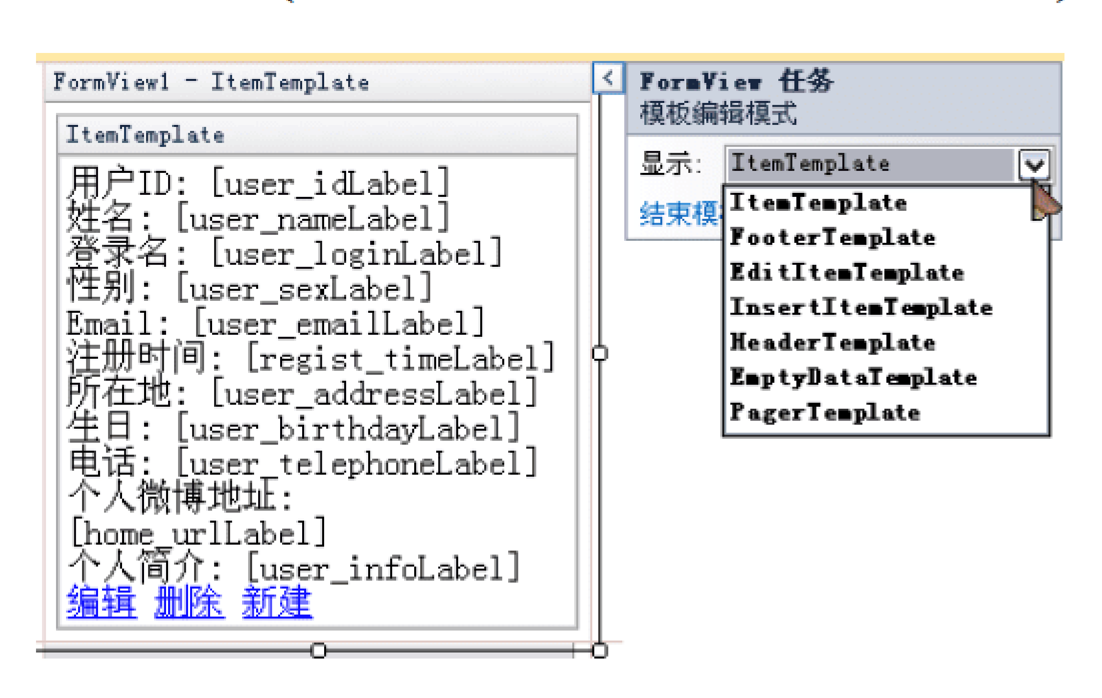


图 6-14　编辑模板模式

(9) 在“属性”面板中设置 FormView 控件的 DataKeyNames 属性为 user_id。

(10) 选中数据源控件 EntityDataSource2，在“属性”面板中单击 Where 属性后面的“…”按钮，打开“表达式编辑器”对话框，在该对话框中选中“基于提供的参数自动生成 Where 表达式”复选框，然后单击“添加参数”按钮，添加参数 user_id，设置“参数源”为 FormView 控件，如图 6-15 所示，生成的数据源控件的代码如下所示。



图 6-15　“表达式编辑器”对话框

```
<asp:EntityDataSource ID="EntityDataSource2" runat="server"
    AutoGenerateWhereClause="True" ConnectionString="name=WeiBoMsgEntities"
    DefaultContainerName="WeiBoMsgEntities" EnableDelete="True"
    EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
    EntitySetName="Z_MESSAGE" Where="" EntityTypeFilter="" Select="">
    <WhereParameters>
        <asp:ControlParameter ControlID="FormView1" Name="user_id"
            PropertyName="SelectedValue" />
    </WhereParameters>
</asp:EntityDataSource>
```

(11) 添加一个 ListView 控件到页面中，设置控件的数据源为 EntityDataSource2 在“ListView 任务”面板上选择“配置 ListView”选项，打开“配置 ListView”对话框，在该对话框中可以选择控件的布局、样式和是否启用插入、更新和分页等操作。布局选择“网格”，样式选择“专业型”，同时选中“启用编辑”、“启用插入”、“启用删除”和“启用分页”复选框，如图 6-16 所示。单击“确定”按钮，关闭该对话框。如果有对话框询问是否想重新生成 ListView 控件，单击“是”按钮即可。

(12) 切换至“源”视图，删除所有模板中的<Z_USER>列，修改 LayoutTemplate 模板中的列标题为中文描述。

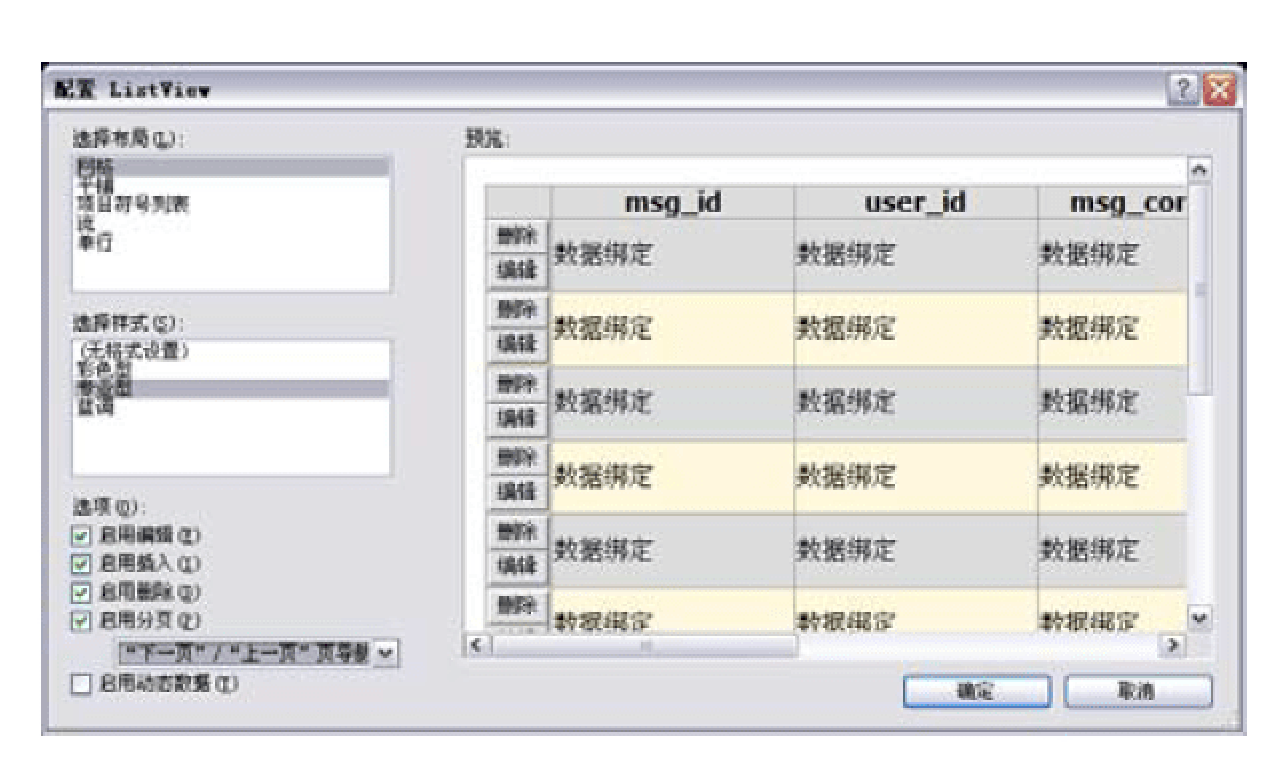


图 6-16　“配置 ListView”对话框

(13) 定位到 LayoutTemplate 标记，找到其中的 DataPager 控件的定义，添加一个 PageSize 特性并设置其值为 3。代码如下所示：

```
        <td runat="server"
            style="">
            <asp:DataPager ID="DataPager1" runat="server" PageSize="3">
                <Fields>
                    <asp:NextPreviousPagerField ButtonType="Button"
ShowFirstPageButton="True"
                        ShowLastPageButton="True" />
                </Fields>
            </asp:DataPager>
        </td>
```

说明：

DataPager 控件的 PageSize 属性默认值为 10，这里为了演示分页功能将其设置的比较小。

(14) 编译并运行程序，页面运行效果如图 6-17 所示。

本例中 FormView 控件支持插入、编辑、删除和分页功能，通过页码选择不同的用户，下面的 ListView 控件将显示该用户发表的信息。ListView 控件也支持插入、编辑、删除和分页功能。

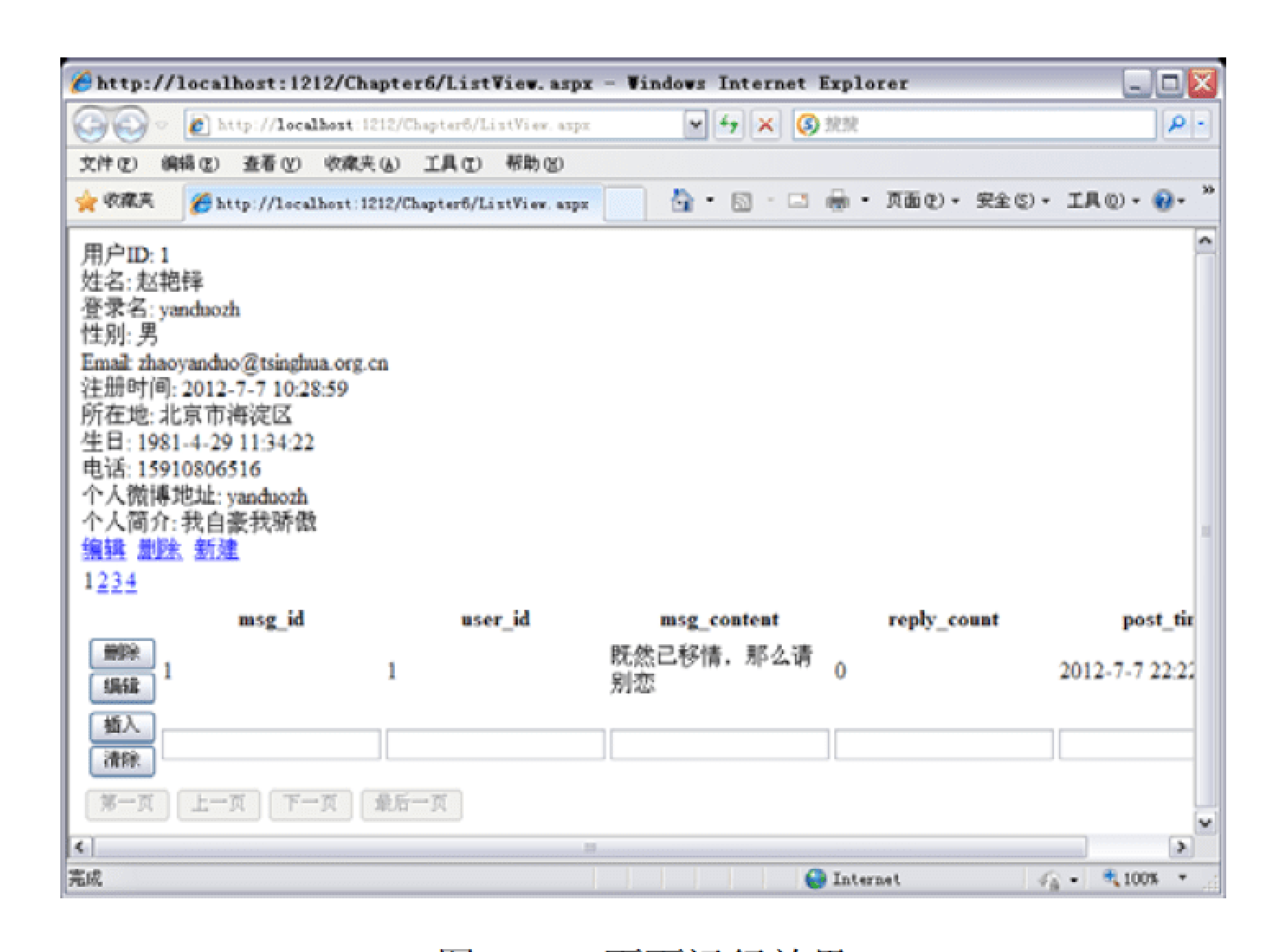


图 6-17　页面运行效果

6.5　本章小结

LINQ 是.NET Framework 4 自带的一项非常有用的技术。它在许多数据访问场景中都是一种重要的管道技术。LINQ 可用于查询内存中的集合。另外，它还可用于 XML、Entities 和 DataSet，这些类型提供了对不同数据存储的访问，但使用相同的、统一的查询语言。本章主要介绍了 LINQ 语言及其语法，以及在 ASP.NET 项目中使用使用 EntityDataSource 控件，这个控件是数据绑定控件和模型之间的桥梁。通过本章的学习，读者应掌握 LINQ 的基本语法以及 LINQ to EF 的具体应用。

6.6　思考和练习

1. 要使用 LINQ，必须引入哪个命名空间？
2. 什么是匿名类，如何定义匿名类？
3. LINQ 查询表达式以什么开头？

4. Single 操作符有什么作用？

5. 如何添加 ADO.NET 实体数据模型。

6. EntityDataSource 数据源控件有什么用？

7. 与其他数据控件(如 GridView 和 Repeater)相比，ListView 控件的主要优势在哪里？

8. 在例 6-5 的基础上，编辑 FormView 控件的 EditItemTemplate 模板和 InsertItemTemplate 模块，实现在编辑和插入记录时，可以通过上传控件来上传头像。

第7章　Web站点中的安全性

一个网站的所有资源通常不是对所有用户都开发，因此，需要考虑用一种安全策略来阻止不受欢迎的用户访问特定内容。还需要考虑一种机制，允许用户注册新账户并登录网站。ASP.NET 4 包含了创建可靠而且安全的安全机制所需的所有工具。本章将介绍 ASP.NET 网站安全性相关的知识，ASP.NET 4 提供的登录控件的使用以及如何使用 ASP.NET 网站管理工具对网站进行安全管理。

本章学习目标：

- ASP.NET 验证方式
- ASP.NET 应用程序服务
- 登录控件的使用
- 使用 WSAT 管理用户和角色
- 以编程方式检查角色

7.1　安全性概述

随着 Web 2.0、社交网络和微博等一系列新型的互联网产品的诞生，Web 环境的互联网应用越来越广泛，企业信息化的过程中各种应用都架设在 Web 平台上，Web 业务的迅速发展也引起黑客们的高度关注，接踵而至的就是 Web 安全威胁的凸显。

7.1.1　关于安全性

安全性是系统设计、实现和管理的一部分，其作用是保证系统可以完全按照人们想要的方式运行。从另一个角度来说，安全性的作用是防止系统被攻击或恶意破坏。

安全性是一个非常复杂的主题，它通常涉及 3 个重要的问题。

- 身份：身份表示用户是谁，对于一个 Web 站点，用户的身份通常是一个登录名。
- 身份验证(Authentication)：身份验证就是如何证明你所具有的身份，通常以用户名和密码配对来确认用户的身份。还有许多其他身份验证机制，包括高科技的指纹或虹膜扫描、智能卡和令牌等。本章只讨论用户名和密码身份验证方式。
- 授权(Authorization)：授权是指登录成功后，用户被允许执行的操作。

ASP.NET 中的这些安全概念是使用应用程序服务(application service)实现的。

7.1.2　ASP.NET 验证方式

ASP.NET 提供多种方式来处理验证请求和确认用户身份，包括 Windows 身份验证、窗体基本验证和护照验证 3 种。

- Windows 身份验证：Windows 身份验证是使用传统“基本”、NTLM/Kerberose 和 Digest 验证方式，它是直接使用 Windows 域用户权限的，换句话说，IIS 服务器直接使用 Windows 用户作为网站会员，通常是使用在 Intranet 环境。
- 窗体基本验证：窗体基本验证方式是使用 Web 窗体获取用户名和密码后，以 FormsAuthentication 类或 Membership 类方法来检查用户身份，可以使用 web.config 文件、XML 文件或数据库来存储会员数据。
- 护照验证：护照验证是 Microsoft 提供的单一登录服务，如同护照一样，用户只需登录一次，就可以进入任何参与此服务的群组网站。

ASP.NET 的验证方式可以通过配置文件 web.config 中的<authentication>和<authorization>标记来设置。

1. <authentication>标记

ASP.NET 建立的 Web 应用程序如果需要使用验证服务，可以在 web.config 文件的<authentication>标记(在<system.web>子标记中)中指定验证方式，<authentication>标记使用 mode 属性指定验证方式，其属性枚举值如表 7-1 所示，例如：

```
<authentication mode="Windows"></authentication>
```

表 7-1　mode 属性的属性值

属　性　值	描　　述
Forms	使用窗体基本验证方式
Windows	使用 Windows 验证方式
Passport	使用护照验证方式
None	不使用验证

2. <authorization>标记

当用户身份经过验证后，就可以授予用户权限。在<authorization>标记(在<system.web>子标记中)中指定用户权限，在该标记中可以有 0 到多个<deny>和<allow>子标记：<allow>标记表示允许存取的资源；<deny>标记表示不允许存取的资源。这两个标记具有相同的属性，如表 7-2 所示。

表 7-2　<allow>和<deny>标记的属性

属　性	描　述
users	使用逗号“，”分隔的用户，这些用户允许或不允许存取资源，“？”代表匿名用户，“*”代表所有用户
roles	使用逗号“，”分隔的角色，默认为 Windows 群组名称，属于该角色的用户允许或不允许存取资源
verbs	使用逗号“，”分隔的 HTTP 传输方法，可以有：GET、HEAD、POST 和 DEBUG

例如，下面的配置表示用户 gemm 允许使用 GET 方法，具有 Administrator 角色的用户允许使用 POST 方法，所有用户都不允许使用 DEBUG 方法。

```
<authorization>
   <allow users="gemm" verbs="GET"/>
   <allow roles="Administrator" verbs="POST"/>
   <deny users="*" verbs="DEBUG"/>
</authorization>
```

3. 窗体基本验证

ASP.NET 窗体基本验证也称 Cookie 基本验证，因为它是使用 Cookie 来存储验证记录的。

当用户成功登录网站后，以 Cookie 存储的验证数据就会建立，用户可以拥有权限进入其他需要验证的网页。如果用户尚未登录，则会进入一个需要验证的页面，就会自动跳转到登录网页，要求用户首先登录网站。

当<authentication>标记的 mode 属性为 Forms 时，就是使用窗体基本验证，其子标记<forms>可以指定验证的 Cookie 名称和登录网页的 URL 地址等相关信息，其相关的属性如表 7-3 所示。

表 7-3　<forms>标记的属性

属　性	描　述
name	指定验证的 Cookie 名称，默认值为.ASPXAUTH，需要指定唯一的 Cookie 名
loginUrl	指定登录网页的 URL 地址，当用户尚未通过验证就转到此页，默认值为 Default.aspx
protection	指定 Cookie 数据的保密方式，可以使用验证或加密保护，属性值 None 表示不保密，Encryption 表示 Cookie 数据需要加密，Validation 表示需要验证，默认值 ALL 表示需要验证和加密
timeout	指定 Cookie 数据的超时时间，以分钟为单位，默认值为 30 分钟
path	指定 Cookie 的路径，默认值为“\”

窗体基本验证的会员数据可以存储在 web.config、XML 文件或数据库中，如果存储在 web.config 文件中，可以在<forms>标记的<credentials>子标记中定义用户名和密码，例如：

```
<credentials passwordFormat="Clear">
   <user name="gemm" password="ilovezhaoyd"/>
   <user name="zhaoyd" password="99123"/>
</credentials>
```

<credentials>标记的 passwordFormat 属性用于指定密码是否需要加密保护，属性值 Clear 表示不加密，MD5 表示使用 MD5 哈希运算加密，SHA1 表示使用 SHA1 哈希运算加密。

<credentials>标记中的每一个<user>子标记，用于指定一位用户的登录名和密码。name 属性指定登录名，password 属性用于指定登录密码。

7.1.3　ASP.NET 应用程序服务

ASP.NET 2.0 之前的版本都有某种安全性支持。不过，它们都缺乏 ASP.NET 2.0 和之后的版本中提供的高级控件和概念。在 ASP.NET 1.x 应用程序中，需要编写大量代码来实现可靠的安全策略。而且需要在所有 Web 站点中重复编写几乎一样的代码。

而自从应用程序服务随着 ASP.NET 2.0 一起问世，这些问题就得到了解决。应用程序服务是可以在 Web 应用程序中用来支持用户、角色和配置文件等的一组服务。ASP.NET 4 中仍然大量存在这类服务。其中最重要的是如下几个。

- 成员(Membership)：可以管理和使用系统中的用户账户。
- 角色(Role)：可以管理用户被指派的角色。
- 配置文件(Profile)：允许将用户特定的数据存储到后台数据库。

图 7-1 列出了这些服务的概览，显示了它们如何与应用程序 Web 站点和服务可能使用的底层数据存储相关。

在图 7-1 的顶部，是 Web 站点和 Web 应用程序。这些网站可以包含控件，如可与 ASP.NET 应用程序服务(如成员和配置文件)通信的登录控件。为了创建一个灵活的解决方案，这些服务不直接与底层数据源通信，而是和配置好的提供者(provider)通信。提供者是软件中可用于特定任务的可交换部分。例如，在成员服务中，成员提供者设计为对底层数据存储中的用户起作用。可以根据需要为相同的应用程序服务配置不同的提供者。

每个提供者都需要一个数据存储(图 7-1 的底部)。每个提供者都编写为与特定数据存储一起使用。例如，SQL Server 成员提供者(用于处理例如创建用户、登录和重置密码这样的成员服务)和 SQL Server 角色提供者(用于处理与角色相关的任务)设计为与 Microsoft SQL Server 数据库一起使用。

在理想情况下，无须直接处理这些提供者。一般情况下，在一个集中的位置为 Web 站点配置不同的提供者。可以通过与应用程序服务通信来使用这些提供者。尽管可以直接从代码访问这些服务，但通常是使用 ASP.NET 内置的登录控件来完成这些困难的工作。

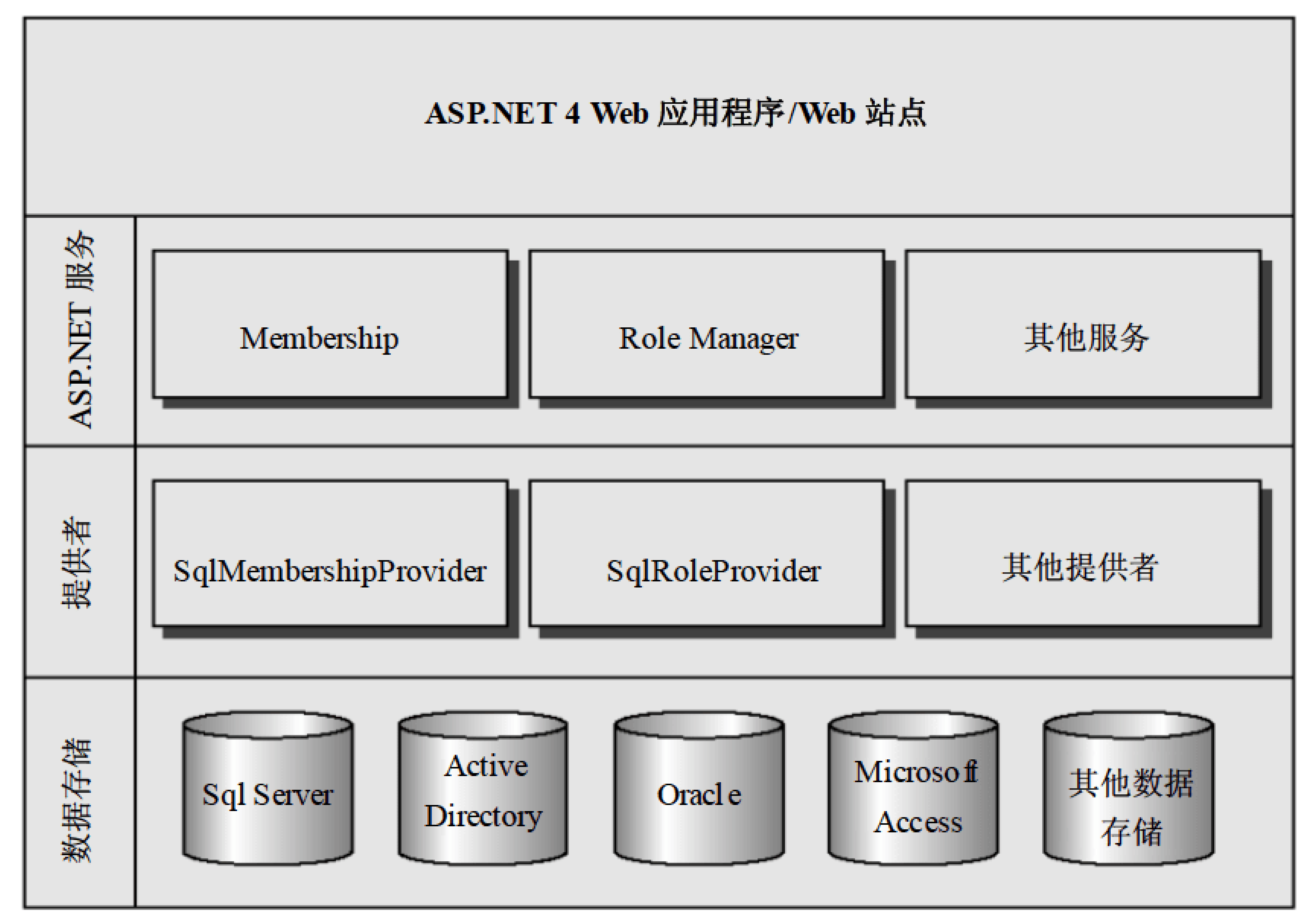


图 7-1　ASP.NET 应用程序服务体系

7.2　登录控件

与数据和导航控件一样，登录控件也是在 ASP.NET 2.0 中引入的，并且在 ASP.NET 4.0 中增加了很多强大的功能。随 ASP.NET 4.0 提供的登录控件去除了大量与编写 Web 站点中的安全层相关的复杂性。可用的登录控件有效地封装了验证和管理用户所需的所有代码和逻辑。这些控件通过应用程序服务，与已配置的提供者通信来进行工作，而不是直接与数据库通信。

ASP.NET 4.0 中包含 7 个登录控件，每个都有不同的用途。有了登录控件，只需很少的工作量就可以构建安全的 Web 站点，另外，还提供了一些工具，让用户可以修改密码；或者如果忘记密码，则可以请求一个新的密码，并允许根据登录状态和用户的角色显示不同的数据。

7.2.1　Login 控件

Login 控件允许用户登录到站点。而在后台，它通过应用程序服务与配置好的成员提供者进行通信，查看用户名和密码是否代表系统中的有效用户。如果用户通过验证，就生成发送到用户浏览器的 cookie。对于后续的请求，浏览器重新提交该 cookie 给服务器，这样系统就知道它仍在处理有效用户。成员提供者的不同设置都在 web.config 文件的 <membership />元素中进行配置。

Login 控件的常用属性如表 7-4 所示。

表 7-4　Login 控件的常用属性

属　　性	描　　述
DestinationPageUrl	该属性定义了登录请求成功后将用户发往哪个 URL
CreateUserText	该属性控制用于邀请用户注册新账户的文本
CreateUserUrl	该属性控制用户注册新账户的页面的 URL
DisplayRememberMe	该属性指定控件是否显示 Remember Me 选项。如果为 False 或在登录时未选择该复选框，那么每次关闭和重新打开浏览器时，用户需要重新进行身份验证
RememberMeSet	该属性指定最初是否选择 Remember Me 选项
PasswordRecoveryText	该属性控制用于告诉用户可重置或恢复他们密码的文本
PasswordRecoveryUrl	该属性指定用户可获取他们(新)密码的页面的 URL
VisibleWhenLoggedIn	该属性控制当前用户登录时控件是否可见。默认值为 True

除了这些属性，Login 控件还有一些 Text 属性，如 LoginButtonText、RememberMeText、TitleText 和 UserNameLabelText，这些属性用于设置控件中和其子控件(如组成用户界面的 Button 和 Label 控件)上出现的文本信息。

默认情况下，ASP.NET 的身份验证机制假定在站点的根目录下有一个用于用户登录的页面 Login.aspx。这个页面要想生效，至少需要有一个 Login 控件。如果要使用不同的页面，可以在<authentication />下面的<forms />元素中指定其路径，如下所示：

```
<authentication mode="Forms">
  <forms loginUrl="MyLoginPage.aspx" />
</authentication>
```

注意：

在 Login 页面(配置在 loginUrl 中)上，Login 控件的 VisibleWhenLoggedIn 属性没有作用。在配置过的 Login 页面上，Login 控件总是可见。如果要隐藏它，可以使用 LoginView 控件。

Login 控件也提供了一些事件，这些事件通常不需要进行处理，但时常都会派上用场。例如，LoggedIn 事件在用户刚登录后触发，如果 DestinationPageUrl 不是很灵活，这里是将用户动态发送到另一页面的理想场所。

例 7-1：使用 Login 控件登录。

(1) 启动 VWD 2010，新建空网站 Chapter7。

(2) 通过“添加新项”对话框添加一个名为 Login.aspx 的页面。

(3) 从“工具箱”的“登录”类别中拖放一个 Login 控件到该页面上。

(4) 打开 Login 控件的“属性”面板，设置控件的 CreateUserText 属性为“注册新用户”、CreateUserUrl 属性为~/Register.aspx、PasswordRecoveryText 属性为“忘记密码”、PasswordRecoveryUrl 属性为“~/PasswordRecovery.aspx”。

当学习其他控件时将创建注册页面和忘记密码页面。

(5) 返回到 Login.aspx，保存所有修改，编译并运行程序，在浏览器中打开 Login 页面。如图 7-2 所示。

(6) 输入任意用户名和密码尝试登录，将显示登录失败，因为此时还没有创建该账户。

说明：

第一次运行该页面时，可能需要一些时间才能看到结果，因为 ASP.NET 忙于建立成员数据库。

(7) 由于此时，注册页面和忘记密码页面还没创建，所以单击相应的链接，将提示页面不存在。

(8) 返回到 VWD 2010，在“解决方案资源管理器”窗口中右击项目名称，从弹出的快捷菜单中选择“刷新文件夹”命令，可以看到，项目中添加了 App_Data 文件夹，其中包括一个数据库文件 ASPNETDB.MDF，如图 7-3 所示。



图 7-2　登录页面

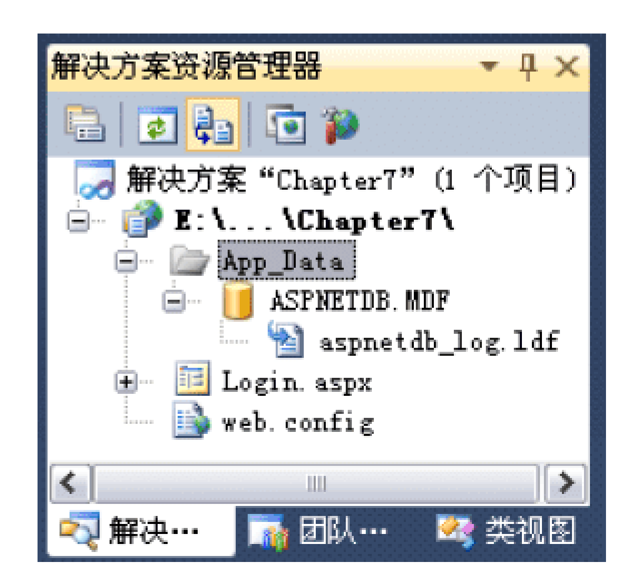


图 7-3　系统自动创建数据库

在本例中，除了添加和配置 Login 控件外，并未编写任何代码。不过，仍能实现一个功能完全的登录过程，在首次尝试登录(或使用需要数据库访问的其他登录控件)时，提供者检查应用程序是否在使用带有必要数据库对象(如表)的数据库。默认情况下，通过查找名为 LocalSqlServer 的连接字符串检查数据库。这个连接字符串在 web.config 文件中无法找到，因为它定义在.NET Framework 文件夹(C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\Config)中名为 machine.config 的文件中。本书也将学习该文件的更多内容。该文件中的连接字符串类似于如下所示：

```
<connectionStrings>
    <add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

这是一个针对 SQL Server 2008 Express Edition(Microsoft SQL Server 的免费版本)的连接字符串。其中|DataDirectory|指向 Web 站点的 App_Data 文件夹。要了解 ASPNETDB.MDF 的详细信息，可以在“服务器资源管理器”窗口中打开该数据库，查看该数据库中的所有表信息。

7.2.2　LoginView 控件

LoginView 控件用于向不同的用户显示不同的数据。它可以区分匿名用户和登录用户，甚至区分不同角色中的用户。LoginView 是模板驱动的，因此程序员可允定义显示给不同用户的不同模板。表 7-5 列出了两个主要的模板和特殊的 RoleGroups 元素。

表 7-5　LoginView 控件的主要模板和 RoleGroups 元素

模　板	描　述
AnonymousTemplate	该模板中的内容只显示给未进行身份验证的用户
LoggedInTemplate	该模板中的内容只显示给登录用户。该模板与 AnonymousTemplate 互斥。任何时候只有其中一个模板可见
RoleGroups	该模板可以包含一个或多个 RoleGroup 元素，这些元素包含一个定义特定角色的内容的 ContentTemplate 元素。允许查看内容的角色定义在 Roles 特性中，该特性采用一个逗号分隔的角色列表。RoleGroups 元素与 LoggedInTemplate 互斥。这意味着如果用户是 RoleGroup 的其中一个角色中的成员，那么 LoggedInTemplate 中的内容就不可见。另外，只有匹配用户角色的第一个 RoleGroup 的内容可见

除了定义在控件的各种子元素中的内容，LoginView 控件本身并不输出任何标记，这意味着可以很容易得将它嵌入一对 HTML 标记之间，如<h1>和<li>，从而创建自定义标题或者列表项。

下列代码段显示了一个 LoginView 控件，它为 3 个不同的用户定义了内容，即访问站点的匿名用户、登录用户，以及已经登录并且是 Managers 角色的成员的用户：

```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
    这是匿名用户看到的信息，是否<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/Register.aspx">注册</asp:HyperLink>为小石头网站的新用户。
    </AnonymousTemplate>
    <LoggedInTemplate>
    这是登录会员看到的信息，欢迎你！
    </LoggedInTemplate>
    <RoleGroups>
        <asp:RoleGroup Roles="Managers">
            <ContentTemplate>
            管理员你好，你可以对本站成员进行管理。
            </ContentTemplate>
        </asp:RoleGroup>
    </RoleGroups>
</asp:LoginView>
```

本章 7.3.2 节将介绍如何创建和配置角色的内容。

7.2.3　LoginStatus 控件

LoginStatus 控件提供了有关用户当前状态的信息。当用户未进行身份验证时，它提供“登录”链接，当用户登录后，它提供“注销”链接。通过设置 LoginText 和 LogoutText 属性，可以控制实际显示的文本。也可以设置 LoginImageUrl 和 LogoutImageUrl 属性显示一个图像而非文本。LogoutAction 属性可用来决定在用户注销时是否刷新当前页面，或是否在用户注销后将用户带至另一个页面，通过设置 LogoutPageUrl 可以确定这一目标页面。

除了这些属性，该控件可以引发两个事件：LoggingOut 和 LoggedOut，它们分别在用户刚注销前后触发。

7.2.4　LoginName 控件

LoginName 是一个极为简单的控件。它的作用就是显示登录用户的名称。为了将用户名嵌入到一些文本中，可以使用 FormatString 属性。例如将 FormatString 属性设置为“当前登录用户是：{0}”，运行时{0}将被实际登录的用户名所取代。

例 7-2：使用 LoginView、LoginStatus 和 LoginName 控件创建网站首页 Index.aspx。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 通过“添加新项”对话框添加名为 Index.aspx 的页面。

(3) 切换到 Index.aspx 的“设计”视图，添加一个 LoginView 控件和一个 LoginStatus 控件。

(4) 选中 LoginView 控件，单击右上角的按钮，打开“LoginView 任务”面板，“视图”列表中显示当前视图为 AnonymousTemplate，表示当前显示为“匿名用户看到的信息，此时在 LoginView 控件上输入一些提示信息，并添加一个 HyperLink 控件，该控件将导航用户到“注册”页面。如图 7-4 所示。

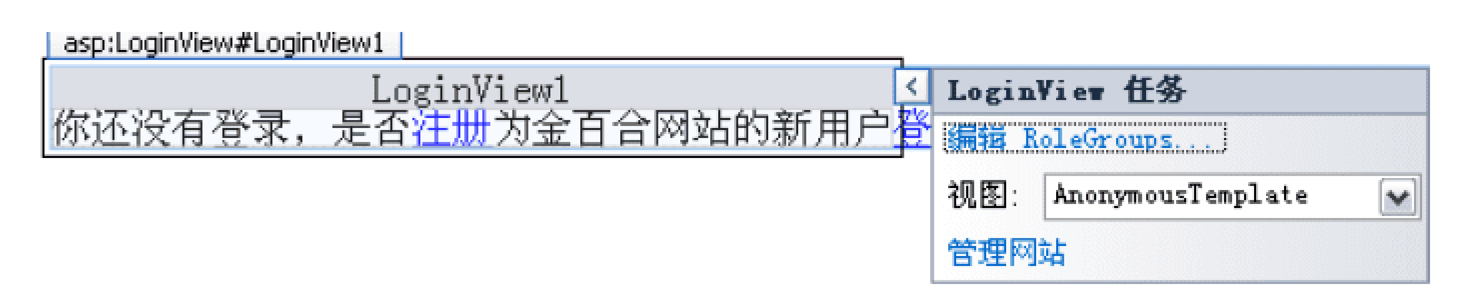


图 7-4　设置 LoginView 控件的 AnonymousTemplates 视图

切换到“源”视图，可以看到生成的代码，如下所示：

```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
        你还没有登录，是否<asp:HyperLink ID="HyperLink1" runat="server"
            NavigateUrl="~/Register.aspx">注册</asp:HyperLink>为金百合网站的新用户
    </AnonymousTemplate>
</asp:LoginView>
```

(5) 在“LoginView 任务”面板中选择 LoggedInTemplate 视图，在 LoginView 上设置登录成功的用户看到的信息，使用 LoginName 控件显示登录用户名，并使用 HyperLink 控件提供“修改密码”超链接，最终生成的代码如下：

```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
        你还没有登录，是否<asp:HyperLink ID="HyperLink1" runat="server"
            NavigateUrl="~/Register.aspx">注册</asp:HyperLink>为金百合网站的新
用户
    </AnonymousTemplate>
    <LoggedInTemplate>
        <asp:LoginName ID="LoginName1" runat="server" FormatString="欢迎你，{0}" />
        <br />你可以
        <asp:HyperLink ID="HyperLink2" runat="server"
NavigateUrl="~/ChangePassword.aspx">修改密码</asp:HyperLink>
    </LoggedInTemplate>
</asp:LoginView>
```

(6) LoginStatus 控件也有两个视图，如果尚未登录，LoginStatus 控件将提供“登录”链接；如图已经登录成功，LoginStatus 控件将提供“注销”链接。但是如何让“登录”链接跳转到例 7-1 中创建的 Login.axpx 页面呢？这需要修改 web.config 的配置信息。打开 web.config，在< system.web>元素下添加或修改认证模式为如下代码：

```
<authentication mode="Forms">
  <forms loginUrl="Login.aspx" timeout="2880" />
</authentication>
```

(7) 编译并运行程序，在浏览器中打开 Index.aspx 页面，如图 7-5 所示。由于此时尚未登录，所以 LoginView 控件显示<AnonymousTemplate>模板的视图，LoginStatus 控件显示为“登录”链接，单击“登录”超链接，将跳转到例 7-1 中创建的 Login.aspx 页面。

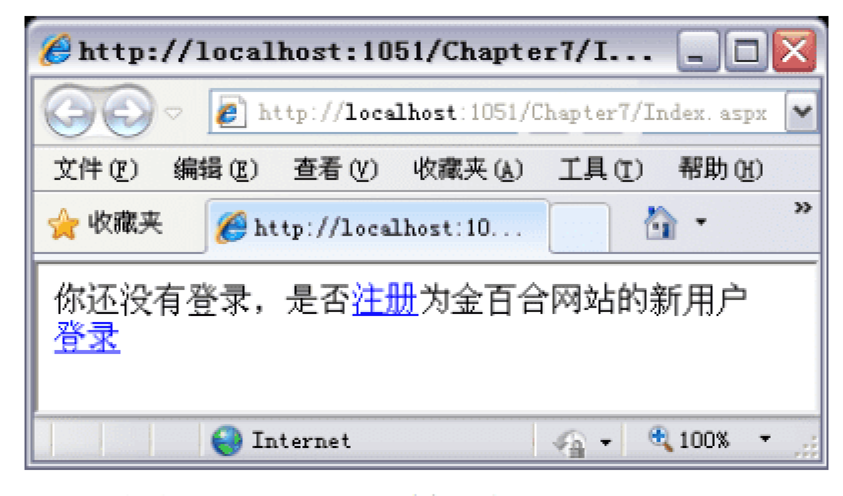


图 7-5　网站首页 Index.aspx

7.2.5　CreateUserWizard 控件

CreateUserWizard 控件用于注册新用户，该控件有一个较长的 Text 属性列表，包括 CancelButtonText、CompleteSuccessText、UserNameLabelText 和 CreateUserButtonText，它

们会影响控件中使用的文本。所有属性都有默认设置，但开发人员也可以将其修改为满足自己需求的内容。

除了文本类属性，该控件还有许多以 ImageUrl 结尾的属性，如 CreateUserButtonImageUrl。这些属性允许定义各种用户动作的图像而非控件生成的默认按钮。如果设置任一属性为有效的 ImageUrl，则还需要设置相应的 ButtonType。例如，要将“创建用户”按钮修改为图像，则需要设置 CreateUserButtonImageUrl 为有效图像并设置 CreateUserButtonType 为 Image。ButtonType 的默认值为 Button，默认情况下它呈现为标准的按钮。也可以设置这些属性为 Link，这样它们将呈现为标准的 LinkButton 控件。

另外，该控件还提供了大量可设置用于修改其行为和外观的有用属性，如表 7-6 所示。

表 7-6　CreateUserWizard 控件的重要属性

属　性	描　述
ContinueDestinationPageUrl	该属性定义用户在注册后单击“继续”按钮时被带往的页面
DisableCreatedUser	当账户创建时是否将用户标记为禁用的。如果设置为 True，那么在账户被启用前，用户不能登录到该站点。默认为 False
LoginCreatedUser	在账户创建后是否让用户自动登录。默认为 True
RequireEmail	决定控件是否要求用户提供电子邮件地址。默认为 True
MailDefinition	该属性包含大量子属性，允许定义在用户注册后发送给他们的电子邮件

CreateUserWizard 控件能够向用户发送一封确认电子邮件，通知它们新账户已经成功建立。这封电子邮件可以提醒它们的用户名和密码。

例 7-3：使用 CreateUserWizard 控件创建注册页面 Register.aspx。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 通过“添加新项”对话框添加名为 Register.aspx 的页面。

(3) 切换到 Register.aspx 的“设计”视图，添加一个 CreateUserWizard 控件。设置控件的 ContinueDestinationPageUrl 属性为~/Index.aspx。

(4) 为了在注册新用户成功后能够给用户发送电子邮件，还需要创建一个邮件模板，在 App_Data 文件夹中添加一个名为 RegistConfirm.txt 的文本文件，文件内容如下：

```
Hi <% UserName %>,

感谢你注册为金百合网站的新用户.
以下是你的用户名和密码信息，请牢记:
用户名:    <% UserName %>
密　码:    <% Password %>

金百合网站全体人员祝你身体健康
```

注意：

在上述文本文件中，占位符 UserName 和 Password 包括在一对服务器端标记(<%和%>)中，当实际发送邮件时，将被用户在注册表单中输入的实际用户名和密码自动取代。

(5) 返回 Register.aspx 页面，在“属性”面板中设置 CreateUserWizard 控件的 MailDefinition 属性，展开该属性，设置 BodyFileName 属性为刚才创建的 RegistConfirm.txt，设置 Subject 属性为“恭喜你成为金百合网站的新用户”，设置 From 属性为网站官方邮箱 t_mse@163.com(用户收到的邮件将显示由该地址发送)。

(6) MailDefinition 属性设置好了，要保证邮件能够正确发送，还需要在 web.config 文件中配置邮件服务器，再次打开 web.config 文件，在根元素<configuration>中添加或修改<system.net>元素，此处需要知道 SMTP 服务器的地址和发送邮件的 Email 地址：

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" >
      <network host="smtp.163.com" />
    </smtp>
  </mailSettings>
</system.net>
```

如果 ISP 要求在发送电子邮件之前进行身份验证或者希望使用一个不同的端口号，可以向<network />元素中添加如下信息：

```
<smtp deliveryMethod="Network">
  <network host="smtp.domain.com" userName="UserName" password="Password" port="587" />
</smtp>
```

一些邮件服务器要求使用 SSL，SSL 是一种加密技术，用于加密发送到邮件服务器的数据以提高安全性能。在 ASP.NET 4 之前的版本中，必须在代码中编程激活 SSL。而 ASP.NET 4 中，<network />元素上有了 enableSsl 特性，如果需要使用 SSL 的邮件服务器，可以使用如下所示的<network />元素：

```
<network enableSsl="true" host="smtp.gmail.com" password="Password" userName="UserName" />
```

(7) 编译并运行程序，在浏览器中打开 Register.aspx 页面，在此输入注册信息，如图 7-6 所示。单击“创建用户”按钮，提示创建成功，如图 7-7 所示。

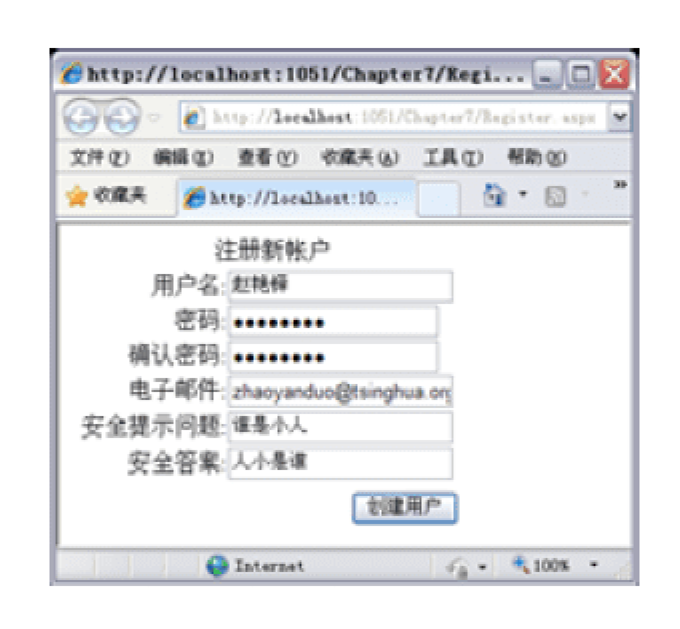


图 7-6　输入注册信息



图 7-7　注册成功

(8) 注册成功后，将在 aspnet_Membership 和 aspnet_Users 表中创建新记录。同时，注册时填写的 Email 中将收到一封由网站发出的邮件，邮件内容就是前面设置的 RegistConfirm.txt 模板，如图 7-8 所示。

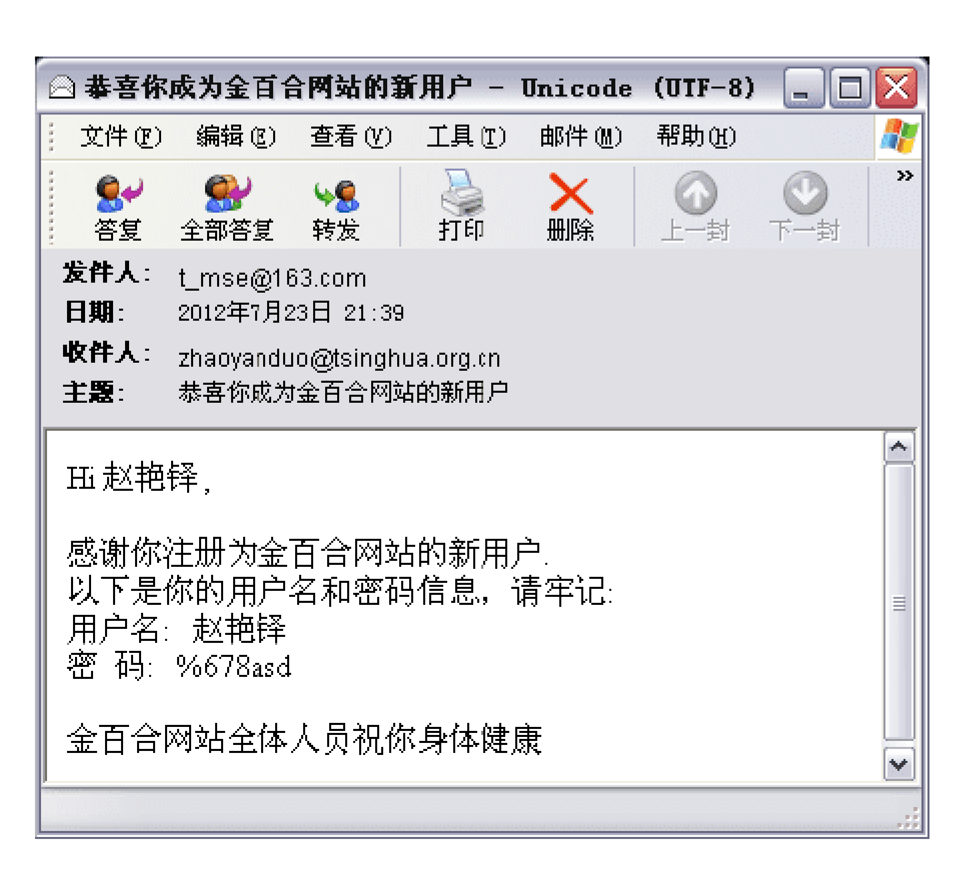


图 7-8　注册成功邮件

其实在注册新用户时，如果使用的密码过于简单，也将提示相应的错误。这些默认配置的来源也是在 machine.config 文件中。打开该文件，定位到<system.web>下面的<membership>元素，如下所示：

```
        <membership>
            <providers>
                <add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider, System.Web, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" connectionStringName="LocalSqlServer"
enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="true"
applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed"
maxInvalidPasswordAttempts="5" minRequiredPasswordLength="7"
minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
passwordStrengthRegularExpression=""/>
            </providers>
        </membership>
```

这就是 ASP.NET 默认的安全策略配置：要求密码长度大于等于 7，且至少包含 1 个数字或特色字符。

如果要为某个应用程序配置不同的安全策略，可以在应用程序的 web.config 文件中修改这些设置。在<system.web>下面添加相应的<membership>元素即可。

默认情况下，密码在数据库中是以散列形式存储的，散列是一个不可逆的过程，它为数据创建唯一索引。因为不可逆，所以没有办法通过散列重新得到原始密码，对于忘记密码的情况，则可以使用 PasswordRecovery 控件生成新的密码，然后将新密码发送到用户注册时填写的 Email 地址中。

7.2.6　PasswordRecovery 控件

PasswordRecovery 控件允许用户获得他们已有的密码(如果系统支持)或是获得一个新的自动生成的密码。在这两种情况下，密码都将发送到用户注册时输入的电子邮件地址中。

PasswordRecovery 控件的大部分属性都是人们所熟悉的。它有大量 Text 属性，如 GeneralFailureText (当密码不可恢复时显示)和 SuccessText，该属性允许设置控件显示的文本。还有一些以 ButtonType、ButtonText 和 ButtonImageUrl 结尾的属性，这些属性允许修改控件的不同动作按钮的外观和行为。如果密码成功恢复，可以设置 SuccessPageUrl 将用户导航到另一个页面。

和 CreateUserWizard 一样，PasswordRecovery 也有一个 MailDefinition 元素，该元素用于指向作为邮件正文发送的文件。可以对用户名和密码使用同样的占位符来自定义消息。如果不对 MailDefinition 进行配置，则控件会使用一个默认的邮件正文。

例 7-4：使用 PasswordRecovery 控件创建忘记密码页面 PasswordRecovery.aspx。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 通过“添加新项”对话框添加名为 PasswordRecovery.aspx 的页面。

(3) 切换到 PasswordRecovery.aspx 的“设计”视图，添加一个 PasswordRecovery 控件，设置控件的 MailDefinition-Subject 属性为“你在金百合网站的新密码”，MailDefinition-From 属性为发送新密码的邮件地址(此地址可以与注册新用户时的地址相同也可以不同)。此处不用设置邮件模板，而是使用默认的邮件模板。

(4) 编译并运行程序，在浏览器中打开 PasswordRecovery.aspx 页面，或者在登录页面时，单击“忘记密码”链接即可打开 PasswordRecovery.aspx，如图 7-9 所示。

(5) 输入“用户名”，单击“提交”按钮将显示注册时填写的“安全提示问题”，输入相应的答案后，单击“提交”按钮，即可收到新密码邮件，如图 7-10 所示。

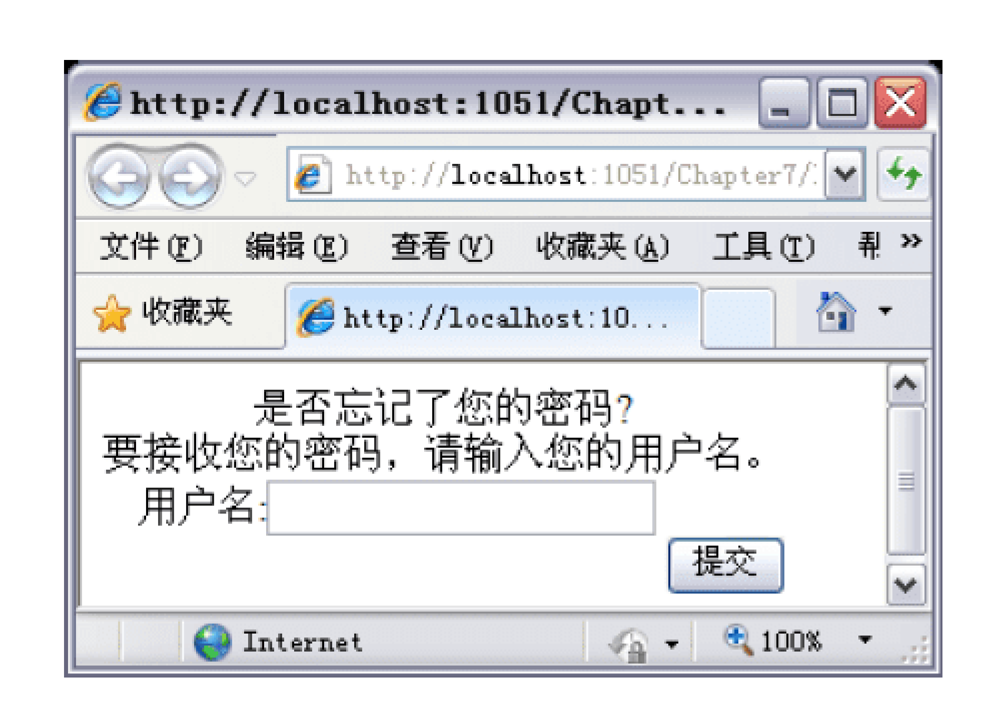


图 7-9　忘记密码页面



图 7-10　网站发送的新密码邮件

7.2.7　ChangePassword 控件

ChangePassword 控件允许已有用户和登录用户更改他们的密码。类似于 CreateUserWizard

和 PasswordRecovery 控件，它有许多属性，可用于修改文本、错误消息和按钮。它也有一个 MailDefinition 元素，允许发送新密码的确认消息给用户的电子邮件地址。

例 7-5：使用 ChangePassword 控件创建修改密码页面 ChangePassword.aspx。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 通过"添加新项"对话框添加名为 ChangePassword.aspx 的页面。

(3) 切换到 ChangePassword.aspx 的"设计"视图，添加一个 ChangePassword 控件，设置控件的 ContinueDestinationPageUrl 和 CancelDestinationPageUrl 属性均为~/Index.aspx，即密码修改成功或取消密码修改都将导航到网站的首页 Index.aspx。

(4) 编译并运行程序，在浏览器中打开 ChangePassword.aspx 页面，或者在登录成功后，单击"修改密码"超链接即可打开 ChangePassword.aspx，如图 7-11 所示。

(5) 在此输入新旧密码，单击"更改密码"按钮，如果密码设置的过于简单，将弹出如图 7-12 所示的错误提示。



图 7-11　修改密码

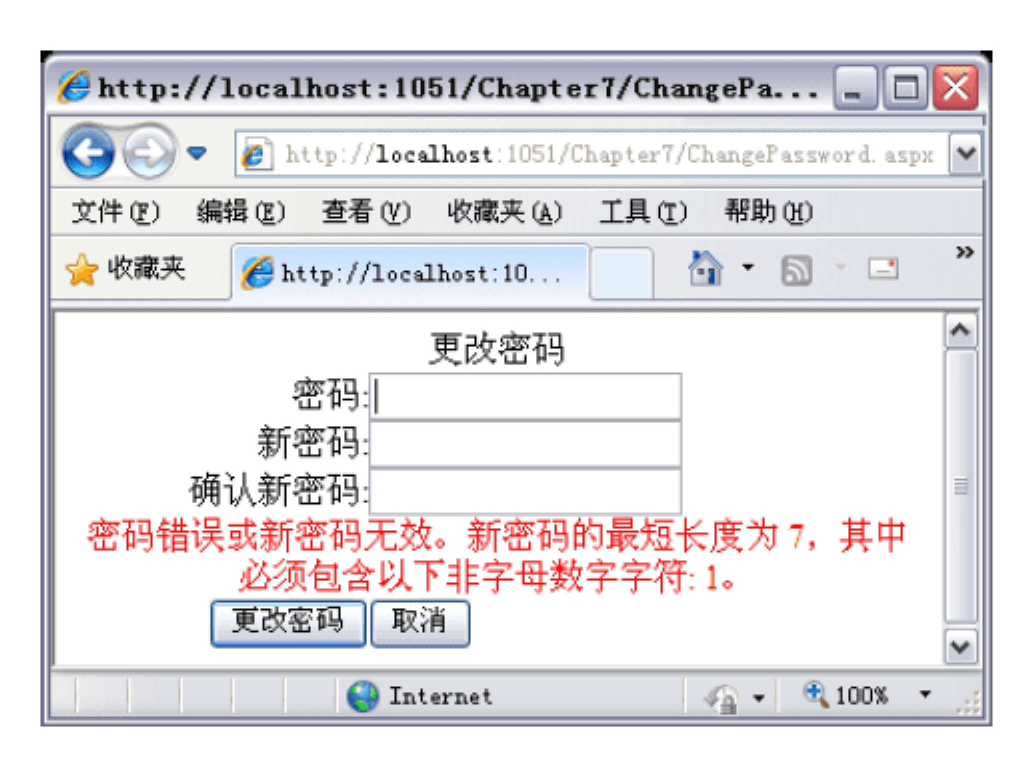


图 7-12　新密码过于简单错误提示

7.3　ASP.NET 网站配置管理

ASP.NET 应用程序服务是可以在 Web 应用程序中用来支持用户、角色和配置文件等的一组服务。通过这些服务可以管理 ASP.NET 的安全、应用程序和提高程序等。

7.3.1　ASP.NET 网站管理工具

ASP.NET 网站管理工具(Web Site Administration Tool，称为 WSAT)是一个选项卡式界面的 Web 工具，该界面将相关的配置设置组合在各个选项卡中。选择"网站"|"ASP.NET 配置"命令，即可打开该工具，如图 7-13 所示。

1. "安全"选项卡

使用"安全"选项卡可以设置和编辑用户、角色和对站点的访问规则，访问规则有助于保证网站内特定资源的安全。

2. “应用程序”选项卡

在此页中可使用不希望硬编码到页面中的值来配置应用程序，使应用程序可以发送电子邮件、配置调试、设置默认错误页，以及停止或启动应用程序。具体包括如下设置。

- 应用程序设置：这些设置是要集中存储并通过代码从网站中的任意位置来访问的名称/值对。
- SMTP 设置：设置网站如何发送电子邮件。
- 应用程序状态：可以设置使网站脱机，以执行维护或使新的 Microsoft SQL Server Standard 版本数据库联机。
- 调试和跟踪：可以设置调试和跟踪选项，还可以定义默认的错误页来报告.aspx 错误。

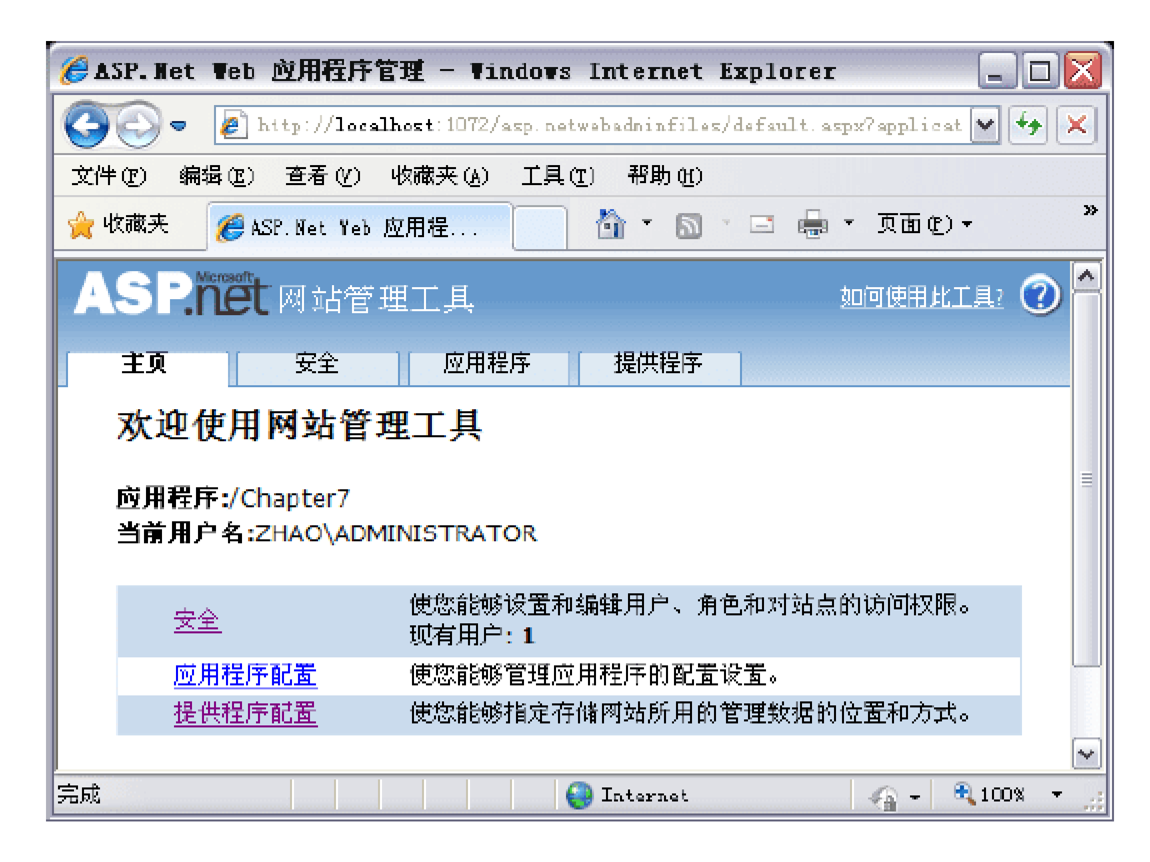


图 7-13　ASP.NET 网站配置管理工具主页面

3. “提供程序”选项卡

使用“提供程序”选项卡可以配置网站管理数据(如成员资格)的存储方式。可以对站点的所有管理数据只使用一个提供程序，也可以为每种功能指定不同的提供程序。

数据库提供程序是为特定功能存储应用程序数据时所调用的类。默认情况下，网站管理工具配置并使用网站的 App_Data 文件夹中的本地 SQL Server Express 数据库。也可以选择使用其他提供程序来存储成员资格和角色管理。

7.3.2　使用 WSAT 管理用户

使用 WSAT 可以完成下列任务：

- 管理用户；
- 管理角色；
- 管理访问规则——如决定什么用户可访问什么文件和文件夹；
- 配置应用程序、邮件和调试设置；
- 使站点脱机，这样用户就不能请求任何页面，而是获得一个友好的错误消息。

使用 WSAT 所作的一些更改将保存在应用程序的 web.config 文件中。而其他一些设置，如创建的用户和角色，存储在已配置的提供者的数据库中。

1. 管理用户和角色

例 7-6：使用 WSAT 管理用户账户和角色。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 选择“网站”|“ASP.NET 配置”命令，打开 ASP.NET 网站管理工具。切换到“安全”选项卡，如图 7-14 所示。



图 7-14　ASP.NET 网站配置管理工具“安全”选项卡

该页的下半部分为 3 个区域：“用户”、“角色”和“访问规则”。在“用户”区域下方有“创建用户”和“管理用户”链接。如果没有这些链接，而是显示一个有关 Windows 身份验证的提示，那么，可以单击“选择身份验证类型”链接，然后选择“通过 Internet”选项，最后单击“完成”按钮，即可得到如图 7-14 所示的页面。

(3) 在“角色”区域，单击“启用角色”链接，将为网站启用角色管理，同时，页面将刷新，出现“禁用角色”和“创建或管理角色”链接。单击“创建或管理角色”链接，打开如图 7-15 所示的“创建新角色”页面。



图 7-15　创建新角色

(4) 输入 Admin 作为新的角色名并单击“添加角色”按钮。这时将出现新的角色。单击右下角的“上一步”按钮，返回主“安全”页面，可以看到“现有角色”数发生了变化。

角色的设置也位于 config 文件中。不过，它默认是不开启的，经过上述操作后，在 web.config 文件的<system.web>元素中将自动添加了如下<roleManager>元素：

```
<roleManager enabled="true" />
```

(5) 单击“用户”区域下方的“创建用户”链接。打开“创建用户”页面，在该页面中输入新用户的详细信息并将该用户指派给 Admin 角色，如图 7-16 所示。

这里输入的用户名是 zhaoyd，密码需要满足之前配置的密码规则，并且选择右侧角色列表中的 Admin 角色名。

(6) 单击“创建用户”按钮将该用户添加到系统中，接着单击确认页面上的“继续”按钮完成用户创建。



图 7-16　创建用户

(7) 单击右下角的“上一步”按钮，返回到主“安全”页面。单击“管理用户”链接。将打开管理用户页面，该页面会显示系统中所有用户，包括前面创建的用户 zhaoyd 和通过 Register.aspx 页面注册的用户，如图 7-17 所示。



图 7-17　管理用户页面

在这里可以启用、禁用、编辑和删除已有用户。通过单击“编辑角色”链接，可以更改指派给用户的角色。使用用户列表上方的筛选控件和字母表，在面对大量用户账户时可以快速搜索特定用户。

(8) 返回 VWD 2010。在“服务器资源管理器”窗口中，打开 ASPNETDB.MDF 数据库，展开“表”节点，右击 aspnet_Roles 表，从弹出的快捷菜单中选择“显示表数据”命令，可以看到前面创建的角色 Admin，用相同的方法打开 aspnet_Membership 和

aspnet_UsersInRoles 表，可以看到创建的用户以及角色和用户的关系。

2. 创建访问规则

创建了用户和角色后，就可以为用户或角色指定特定的权限。

例 7-7：使用 WSAT 创建访问规则。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 打开 ASP.NET 网站管理工具并切换到“安全”选项卡。

(3) 在“访问规则”区域中单击“创建访问规则”链接，打开“添加新访问规则”页面，在此页面中可以设置网站的某个目录的访问规则，允许或者拒绝指定的角色或用户具有该目录的访问权限，如图 7-18 所示。

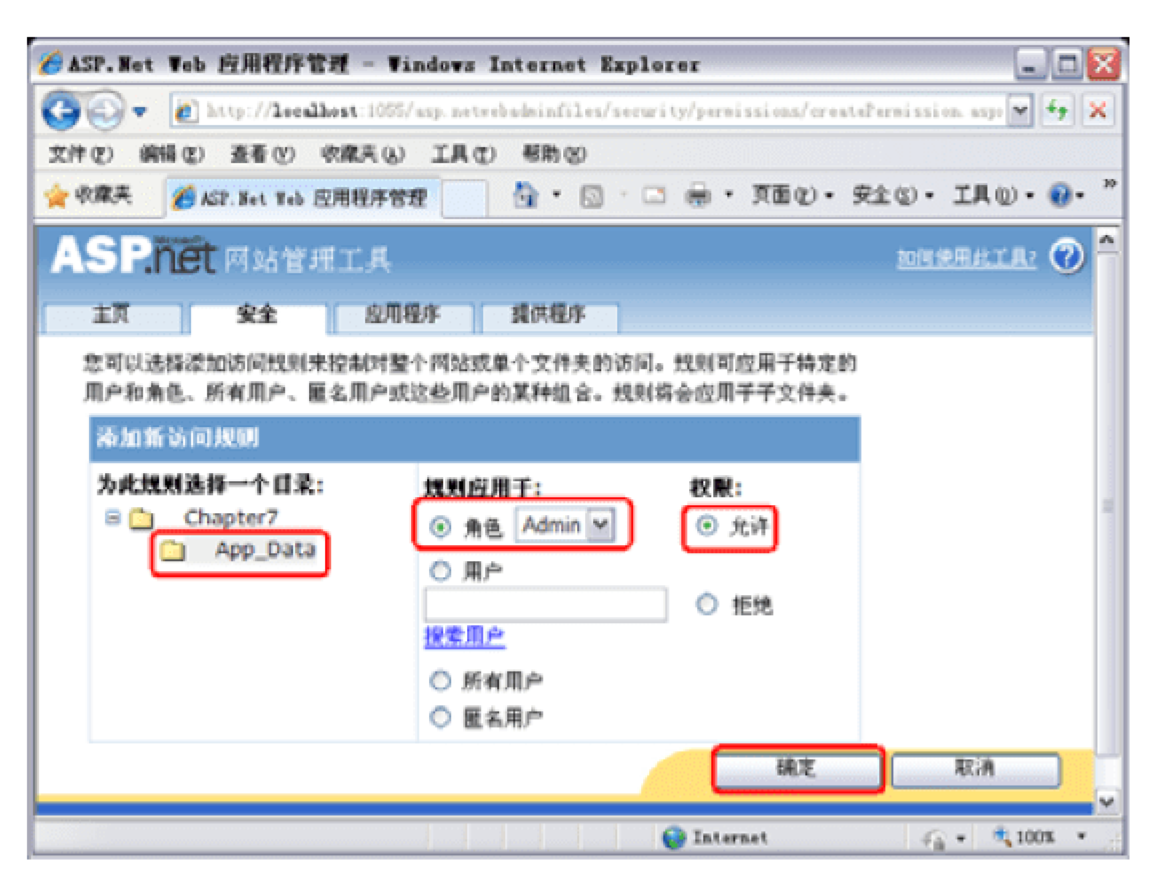


图 7-18　添加新访问规则页面

(4) 单击“确定”按钮即可添加新的访问规则，添加了上述规则后，在相应的子目录下会创建 web.config 文件，并在<system.web>元素下生成相应的规则，示例如下：

```
<system.web>
    <authorization>
        <allow roles="Admin" />
    </authorization>
</system.web>
```

这样创建的访问规则有一个缺点：就是它将安全设置存储在不同的 web.config 文件中，配置的每个子文件夹都有一个。

3. 使用<location>元素

除了通过 WSAT 来添加访问规则以外，ASP.NET 还允许使用<location>元素将相同的设置配置到主 web.config 文件中。<location>元素有一个 path 属性，它可以指向要进行不同配置的文件或文件夹。也可以对 web.config 文件的其他许多(但并非全部)设置使用<location>元素。

例 7-8：使用<location>元素设置目录的访问权限。

(1) 启动 VWD 2010，打开网站 Chapter7。

(2) 在网站根目录下新建一个文件夹 Private。通过<location>元素设置只有 Admin 角色可以访问该目录。

(3) 打开站点根文件夹下的 web.config 文件。在</configuration>结束标记之前，添加一个<location>元素，设置其 path 属性为 Private，然后输入如下配置信息：

```
<location path="Private">
  <system.web>
    <authorization>
      <allow roles="Admin" />
      <deny users="*" />
    </authorization>
  </system.web>
</location>
```

(4) 保存并关闭 web.config 文件。

(5) 打开例 7-2 中创建的 Index.aspx 页面，切换到“设计”视图，选择 LoginView 控件，然后打开其任务面板。在该面板顶部，单击“编辑 RoleGroups”链接，如图 7-19 所示。

(6) 在打开“RoleGroup 集合编辑器”对话框中，单击“添加”按钮插入一个新的成员，然后设置该组的 Roles 属性为 Admin，如图 7-20 所示。

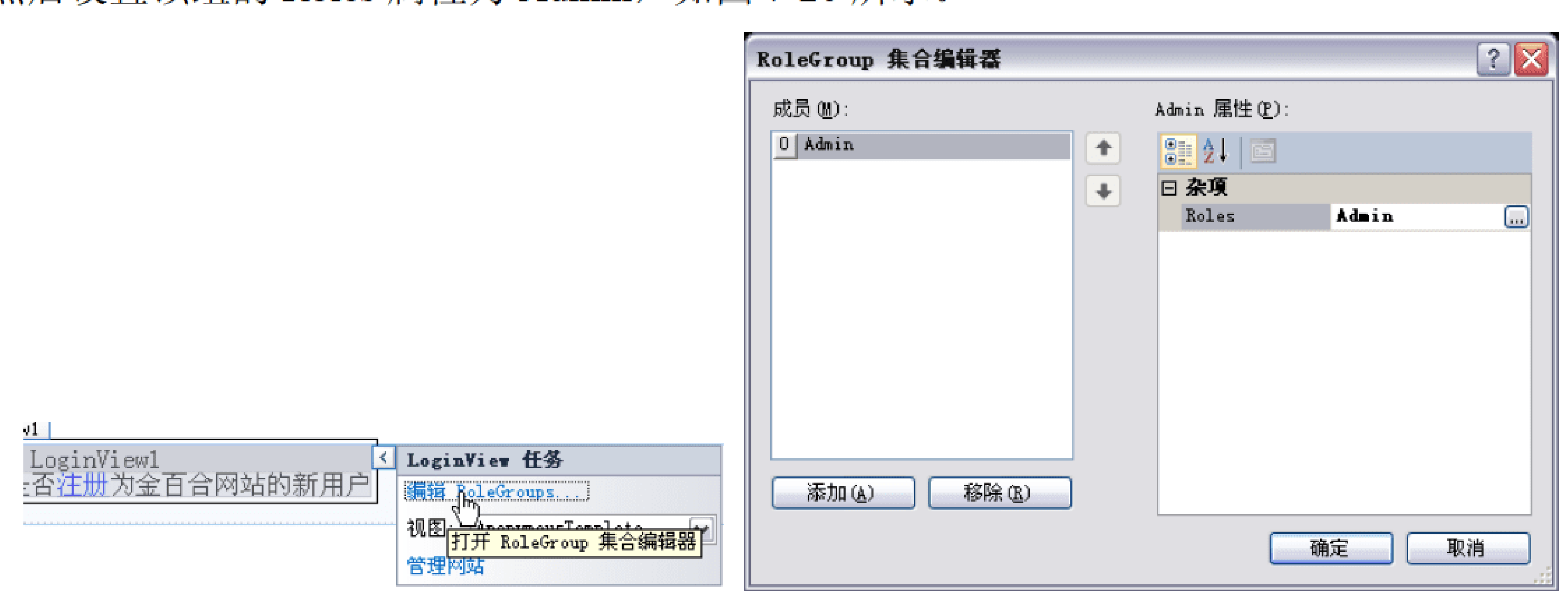


图 7-19　选择“编辑 RoleGroups”选项　　图 7-20　“RoleGroup 集合编辑器”对话框

(7) 单击“确定”按钮，即可创建 RoleGroup，同时返回页面的“设计”视图。

(8) 接下来，在 LoginView 的任务面板中的“视图”下拉列表中将出现 RoleGroup[0]-Admin 选项，如图 7-21 所示。选择该选项，即可编辑相应的模板，添加只对 Admin 角色可见的内容。

(9) 此处添加一个 HyperLink 控件到 LoginView 中，设置该控件的 Text 属性为“管理员站点”，NavigateUrl 属性为“~/Private/Default.aspx”。完成后，生成的 LoginView 控件的代码如下：

```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
        你还没有登录，是否<asp:HyperLink ID="HyperLink1" runat="server"
```

```
                    NavigateUrl="~/Register.aspx">注册</asp:HyperLink>为金百合网站的新
用户
            </AnonymousTemplate>
            <LoggedInTemplate>
                <asp:LoginName ID="LoginName1" runat="server" FormatString="欢迎你，{0}" />
                <br />你可以
                <asp:HyperLink ID="HyperLink2" runat="server"
NavigateUrl="~/ChangePassword.aspx">修改密码</asp:HyperLink>
            </LoggedInTemplate>
            <RoleGroups>
                <asp:RoleGroup Roles="Admin">
                    <ContentTemplate>
                        <asp:HyperLink ID="HyperLink3" runat="server"
NavigateUrl="~/Private/Default.aspx">管理员站点</asp:HyperLink>
                    </ContentTemplate>
                </asp:RoleGroup>
            </RoleGroups>
        </asp:LoginView>
```

(10) 在 Private 文件夹下新建名为 Default.aspx 的页面，也就是 Admin 角色登录可访问的页面，读者可自行设计该页。

(11) 保存所有更改，编译并运行程序，在浏览器中打开站点的首页 Index.aspx，以 Admin 角色的用户登录，即可看到刚才设计的<RoleGroups>模板中的内容，如图 7-22 所示。



图 7-21　视图列表新增了 RoleGroups 选项



图 7-22　Admin 角色用户登录成功后的页面

(12) 单击“管理员站点”链接，可跳转到 Private 目录下的 Default.aspx 页面。

7.3.3　以编程方式检查角色

尽管可以很容易地使用 LoginView 控件在运行时为不同用户展示不同内容，但它并不总是能满足所有要求。有时，还需要根据某人的角色成员资格以编程方式控制显示的数据。

有如下两种方法可以访问当前用户的角色信息。

- 可以通过当前页面或用户控件访问 User 属性的 IsInRole 方法，如下所示：

```
if (User.IsInRole("Admin"))
{
    //do something
}
```

● 可访问包含可直接访问的大量静态方法的 Roles 类。如下所示：

```
if (Roles.IsUserInRole("Admin"))
{
    // do something
}
```

提示：

如上述代码所示，要使用 Roles 类，则必须启用“角色管理”，并在使用该类的代码文件中使用 using 语句引入其所在的命名空间 System.Web.Security。

除了 IsUserInRole 方法，Roles 类还包含了大量其他方法，这些方法允许用户以编程的方式使用角色。例如，可以创建和删除角色、将用户指派给角色或从角色中删除用户，以及获取指派给某一特定角色的用户列表。

7.4　本章小结

本章介绍了网站安全性相关的基础知识，包括 ASP.NET 验证方式、ASP.NET 应用程序服务、登录控件的使用以及 ASP.NET 网站管理工具的使用。ASP.NET 站点中的安全性可以通过一些技术来实现，包括 Windows 身份验证、第三方身份验证和使用 Forms 身份验证。ASP.NET 4 中包含 7 个登录控件，每个都有不同的用途。有了登录控件，只需很少的工作量就可以构建安全的 Web 站点。通过本章的学习，读者应掌握网站安全性相关的基础知识，学会登录控件的使用，并能通过 ASP.NET 网站管理工具设置站点的安全性。

7.5　思考和练习

1. 身份验证和授权的区别是什么？
2. 简述 ASP.NET 的验证方式。
3. LoginView 和 LoginStatus 控件的区别是什么？
4. 如何启用站点的角色管理？

第8章　ASP.NET AJAX

Ajax 是一种应用广泛而且十分有趣的技术，它可以给站点增加许多功能。ASP.NET AJAX 采用异步编程方式，能够实现对客户端脚本的自动管理。利用 ASP.NET AJAX 服务器控件，可以实现局部页面更新的效果。本章首先将介绍 ASP.NET AJAX 的基本知识，然后详细讲解 ASP.NET AJAX 服务器控件的使用方法。通过本章的学习，读者将掌握 ASP.NET AJAX 提供的各种服务器控件的用法，以及使用客户端 ASP.NET AJAX Library。

本章学习目标：

- 了解 ASP.NET AJAX 的基本知识
- 使用 UpdatePanel 控件
- 使用 UpdateProgress 控件通知用户 Ajax 操作的进程
- 使用 Timer 控件
- 使用客户端 ASP.NET AJAX Library

8.1　AJAX 入门

本书已经介绍了浏览器如何与服务器进行交互。浏览器使用 GET 或 POST 方法来请求页面，服务器处理该页面，并回发生成的 HTML 代码。然后，浏览器解析该 HTML 代码并将页面呈现给用户，并有选择地下载任意的外部资源，如图像、脚本文件和 CSS 样式表。当用户与页面交互时(如通过单击按钮来提交一个已填好信息的联系表单)，页面被回发给服务器，之后浏览器会再次加载整个页面。

尽管上述模型已经在 Web 页面中使用很多年了，但是它仍然存在一些缺点。首先，因为整个页面是在回发后被加载的，因此发送到浏览器的 HTML 代码量要远大于浏览器所需要的；加载整个页面的另一个缺点与浏览器显示页面的方式有关，由于整个页面被替换掉，因此浏览器会不得不关闭旧页面，再打开新页面，这样就会使页面“闪烁”，从而使其页面失去对用户的吸引力。Ajax 就是为了解决上述两个问题而产生的。

8.1.1　AJAX 简介

Ajax(Asynchronous JavaScript and XML)技术是由 Jesse James Garrett 提出的，是综合异步通信、JavaScript 以及 XML 等多种网络技术新的编程方式。如果从用户角度看到的实际效果来看，也可以形象地称之为无页面刷新技术。

Ajax 技术的主要内容包括：基于 Web 标准 XHTML+CSS 的表示；使用 DOM(Document

Object Model)进行动态显示及交互；使用 XML 和 XSLT 进行数据交换及相关操作；使用 XMLHttpRequest 进行异步数据检索；使用 JavaScript 将所有的东西绑定在一起等。

Ajax 技术的最大优点是能在不更新整个页面的前提下维护数据。这使得 Web 应用程序可以更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

要使用 Ajax 功能增强 Web 站点，可以选择不同的 Ajax 架构。其中许多架构都能提供一组功能和工具，包括用于在浏览器和服务器端激活 Ajax 的客户端 JavaScript 架构。

8.1.2　ASP.NET AJAX

2005 年，Microsoft 公司在专业开发人员大会上宣布将在 ASP.NET 上实现 Ajax 功能(开发代号为 Atlas)，主要是为了充分利用客户端 JavaScript、DHTML 和 XMLHttpRequest 对象。目的是帮助开发人员创建更具交互性的支持 Ajax 的 Web 应用程序。直到 2007 年 1 月，Microsoft 公司才真正推出了具有 Ajax 风格的异步编程模型，这就是 ASP.NET AJAX 1.0。同时，为了与其他的 Ajax 技术区分，Microsoft 公司用大写的 AJAX，并在其前面加上 ASP.NET。

ASP.NET AJAX 1.0 是以可以在 ASP.NET 2.0 之上安装的单独下载的形式发布的。从.NET Framework 3.5 开始，所有这些特性都成为 ASP.NET 所固有的，这意味着在构建或部署应用时，不再需要下载和安装单独的 ASP.NET AJAX 安装文件。

在 ASP.NET 4 中，它被完全集成在.NET 4 Framework 和 VWD 2010 中，并且与其他客户端架构(包括 jQuery)具有很好的互操作性。

通过 ASP.NET AJAX，开发人员可以实现如下功能。

- 创建无闪烁页面，它们允许刷新部分页面，而不需要全部重载，也不会影响页面的其他部分。
- 在这些页面刷新过程中给用户提供反馈。
- 更新部分页面，使用计时器按计划调用服务器端的代码。
- 访问服务器端 Web 服务和页面方法，使用它们返回的数据。
- 使用富客户端编程架构访问和修改页面中的元素，访问代码模型和典型系统。

ASP.NET AJAX 包括两个重要的部分：ASP.NET AJAX 服务器控件和客户端 ASP.NET AJAX Library。

对于 Web 开发来说，ASP.NET AJAX 从基础框架实现，到客户端与服务器的通信，会发生非常大的变化。相对于 ASP.NET 来说，ASP.NET AJAX 是一种更为成熟的 Web 开发技术。下面将介绍 ASP.NET AJAX 的主要控件 ScriptManager、UpdatePanel、UpdateProgress 和 Timer 的使用方法。

8.2　使用 AJAX 控件

ASP.NET AJAX 完全综合集成到了 ASP.NET 和 VWD 中，这就意味着可以立刻开始使用它。对于在 VWD 中创建的每个新的 ASP.NET 4 Web 站点来说，它们的 Ajax 功能都已

经被激活了。此外，工具箱中包含一个 AJAX Extensions 类别，该类别中包含了要在页面中使用的与 Ajax 相关的控件。VWD 还对 ASP.NET AJAX 提供了大力支持，它为服务器端的控件和客户端的 JavaScript 代码提供了“智能提示”功能。

8.2.1　ScriptManager 控件

ScriptManager 控件是 ASP.NET AJAX 的核心，它提供处理页面上的所有 ASP.NET AJAX 控件(UpdatePanel、UpdateProgress 等)的支持，没有该控件的存在其他 ASP.NET AJAX 控件是不能工作的，并且所有需要支持 ASP.NET AJAX 的 ASP.NET 页面上只能有一个 ScriptManager 控件。此外，ScriptManager 控件还可以生成相关的客户端代理脚本以便能够在客户端脚本中访问 Web 服务。

1. ScriptManager 控件的属性

ScriptManager 类有许多属性，其中大多数都用于高级场景。在很多情况下，不需要改变 ScriptManager 类的任何属性。而在有些情况下，则需要改变或设置它的某些属性。表 8-1 列出了 ScriptManager 控件的一些常见属性。

表 8-1　ScriptManager 控件的重要属性

属　　性	说　　明
AllowCustomErrorsRedirect	该属性确定 Ajax 运行过程中出现的错误是否会导致加载自定义的错误页面。默认为 True；设置为 False 时，错误在浏览器中显示为 JavaScript 通知窗口，或者在禁止调试时对客户端隐藏。注意，如果没有配置任何自定义错误页面，错误就总是显示为 JavaScript 通知
AsyncPostBackErrorMessage	异步回传发生错误时的错误信息。如果没有使用自定义错误页面，这个属性允许自定义错误消息，当发生 Ajax 错误时，用户可以看到这条错误消息
AsyncPostBackTimeout	异步回传时超时限制，默认值为 90，单位为秒
EnablePageMethods	这个属性确定是否允许客户端代码调用页面内定义的方法。后面将讨论其工作原理
EnablePartialRendering	该属性确定 ScriptManager 是否支持使用 UpdatePanel 控件呈现部分页面。除非想阻止整个页面的部分更新，否则应该将它设置为 True
EnableCdn	若该属性设置为 True， ASP.NET 将会包含微软的 Content Delivery Network 站点上(而不是自己的服务器上)的客户端框架文件的链接。这样可以节省一些带宽，如果用户已经从使用这些文件的其他站点那里获取了这些文件的高速缓存副本的话，这样做还可能会提高页面首次加载时的速度
MicrosoftAjaxMode	该属性用于确定是否包含 Microsoft AJAX 客户端库。允许使用 ScriptManager 控件完成与服务器相关的任务(如注册客户端脚本)，而不需要在页面中嵌入客户端框架

(续表)

属　　性	说　　明
ScriptMode	指定 ScriptManager 发送到客户端的脚本的模式，有 4 种模式：Auto，Inherit，Debug 和 Release，默认值为 Auto
ScriptPath	设置所有脚本块的根目录，作为全局属性，包括自定义的脚本块或者引用第三方的脚本块。如果在 Scripts 中的<asp:ScriptReference />标签中设置了 Path 属性，它将覆盖该属性
Scripts	ScriptManager 控件的<Scripts>子元素允许添加客户端在运行时必须下载的其他的 JavaScript 文件
CompositeScript	和<Scripts>元素一样，<CompositeScript>元素也允许添加其他的 JavaScript 文件。但是，在<Composite- Script>元素下注册的文件都被合并为一个单独的、可下载的文件，从而可以减小网络开销并提高页面的性能
Services	<Services>元素允许定义客户端页面能够访问的 Web 服务
OnAsyncPostBackError	异步回传发生异常时的服务端处理函数，在这里可以捕获信息并作相应的处理
OnResolveScriptReference	指定 ResolveScriptReference 事件的服务器端处理函数，在该函数中可以修改某一条脚本的相关信息如路径和版本等

ScriptManager 控件是客户端页面和服务器之间的桥梁。它管理脚本资源(客户端使用的 JavaScript 文件)，负责部分页面的更新(如前所述)以及处理与 Web 站点的交互，如 Web 服务和 ASP.NET 应用程序服务(如成员、角色和配置文件)。

说明：

如果认为只在一小部分页面上需要 Ajax 性能，那么通常可以将 ScriptManager 控件直接放置到内容页中。但是，也可以将 ScriptManager 控件放置在母版页中，这样它便在整个站点中都可用。

2. ScriptManager 控件的用法

要使用 ASP.NET AJAX 提供的功能，必须在网页中包含一个 ScriptManager 控件。添加 ScriptManager 控件后，将生成如下代码：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

在介绍完 UpdatePanel 控件后，将一起举例说明 ScriptManager 控件的用法。

8.2.2　UpdatePanel 控件

UpdatePanel 控件是 ASP.NET AJAX 中最重要的一个控件，它可以用来创建局部更新

的 Web 应用程序。有了 UpdatePanel 控件，开发者不需要编写任何客户端脚本，只需在页面上添加 UpdatePanel 控件和 ScriptManager 控件就可以自动实现局部更新。

1. UpdatePanel 控件的工作原理

UpdatePanel 控件的工作过程如图 8-1 所示。

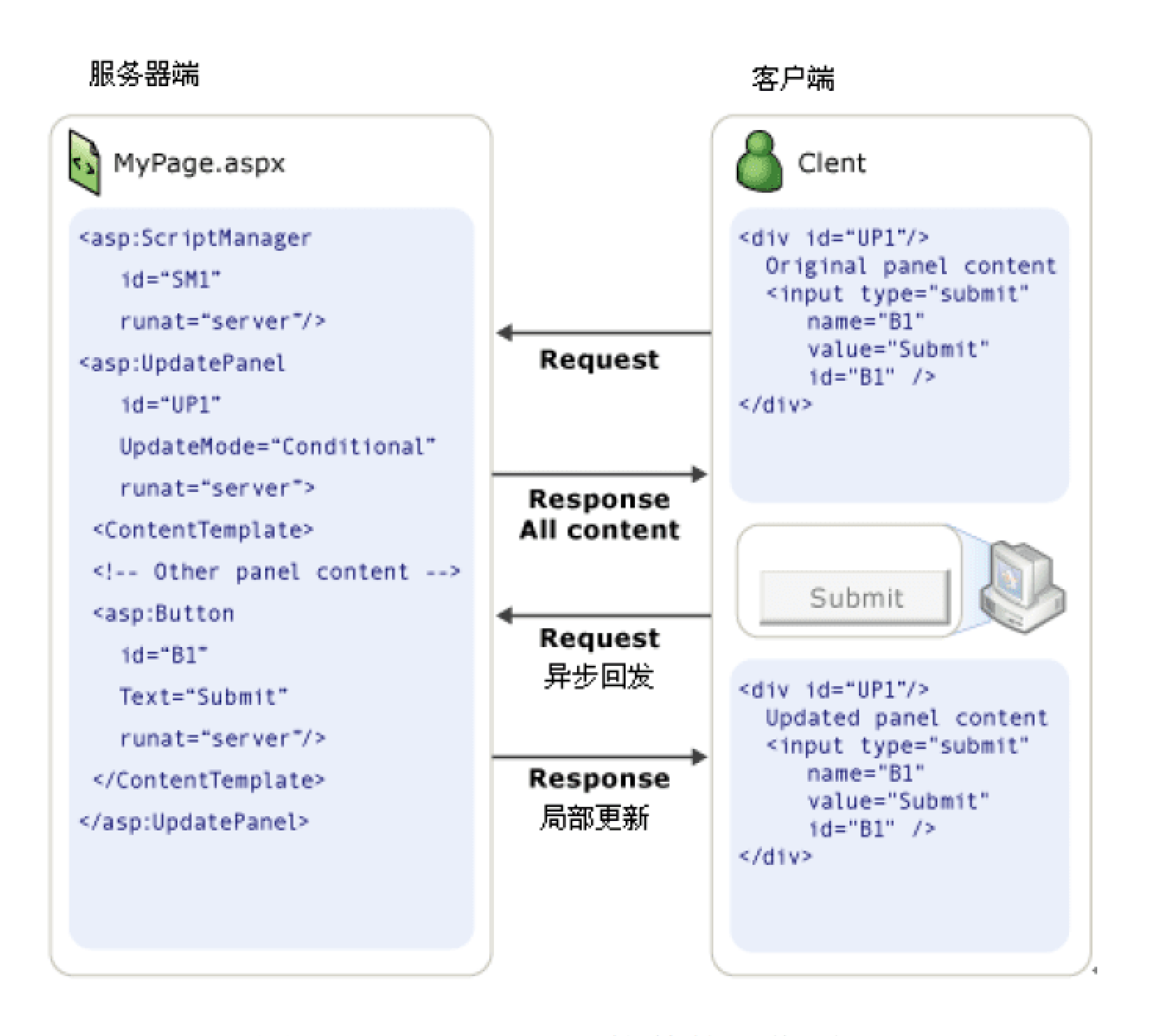


图 8-1　UpdatePanel 控件的工作原理

当客户端向服务器第 1 次发出请求时，服务器返回整个页面。除此之外，均通过异步回传方式对页面进行局部更新。

2. UpdatePanel 控件的属性

UpdatePanel 控件常用属性如表 8-2 所示。

表 8-2　UpdatePanel 控件的重要属性

属　　性	说　　明
ChildrenAsTriggers	该属性确定位于 UpdatePanel 内的控件能否刷新 UpdatePanel。其默认值是 True，当该值设置为 False 时，必须将 UpdateMode 设置为 Conditional。注意，当设置为 False 时，UpdatePanel 内定义的控件仍然会引发到服务器的回发，只是不再自动更新面板
Triggers	Triggers 集合包含 PostBackTrigger 和 AsyncPostBackTrigger 元素。如果要实现完整的页面刷新，那么就可以用第一个；而如果要使用在面板之外定义的控件更新 UpdatePanel，则用第二个
RenderMode	该属性表示 UpdatePanel 最终呈现的 HTML 元素。Block(默认)表示<div>，Inline 表示<span>

(续表)

属　　性	说　　明
UpdateMode	该属性表示 UpdatePanel 的更新模式，有两个选项：Always 和 Conditional。Always 是不管有没有 Trigger，其他控件都将更新该 UpdatePanel，Conditional 表示只有当前 UpdatePanel 的 Trigger 或 ChildrenAsTriggers 属性为 true 时当前 UpdatePanel 中控件引发的异步回送或者整页回送，或是服务器端调用 Update()方法才引发更新
ContentTemplate	该属性在“属性”面板中不可见，但它是 UpdatePanel 的一个重要属性。它是一个容器，用于定义 UpdatePanel 的内容，可以将控件放置在该容器中作为 UpdatePanel 的子控件。如果忘记了这个必需的 ContentTemplate 属性，VWD 会发出一条警告

3. 实现局部更新

在一个页面中，如果需要局部更新的内容较少，可放置一个 UpdatePanel 控件，在该控件内实现局部更新的效果。

例 8-1：在 UpdatePanel 中实现局部更新。

(1) 启动 VWD 2010，新建空网站 Chapter8。

(2) 通过“添加新项”对话框，添加一个名为 AsyncFresh.aspx 的页面。

(3) 切换到页面的“设计”视图，添加一个 ScriptManager 和一个 UpdatePanel 控件。

(4) 在 UpdatePanel 内部添加一个 Label 控件和一个 Button 控件，同时在 UpdatePanel 外面也添加一个 Button 控件。控件的布局如图 8-2 所示。

(5) 切换到“源”视图，相应的代码如下：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <br /><asp:Button ID="Button1" runat="server" Text="UpdatePanel 内异步刷新" />
        </ContentTemplate>
    </asp:UpdatePanel><br />
    <asp:Button ID="Button2" runat="server" Text="UpdatePanel 外" />
</div>
```

(6) 分别为两个 Button 控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 内局部刷新，当前时间：" + DateTime.Now.ToString();
}
protected void Button2_Click(object sender, EventArgs e)
{
```

```
        Label1.Text = "UpdatePanel 外整个页面刷新，当前时间：" + DateTime.Now.ToString();
    }
```

(7) 编译并运行程序，在浏览器中打开 AsyncFresh.aspx，分别单击不同的按钮，观察有什么不同，体验 AJAX 局部刷新的效果，如图 8-3 所示是局部刷新无闪烁的情况。

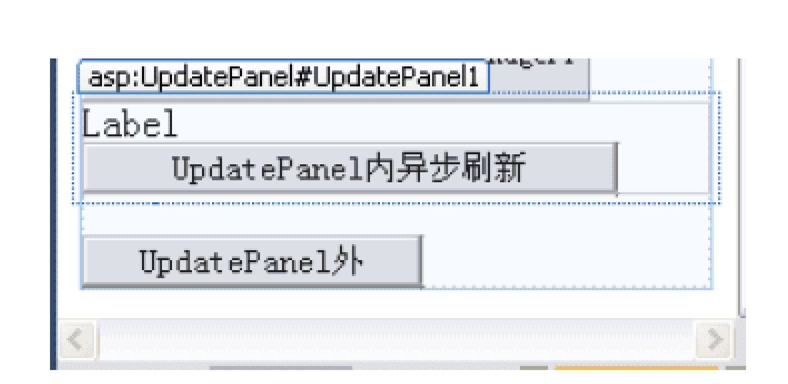


图 8-2　控件布局

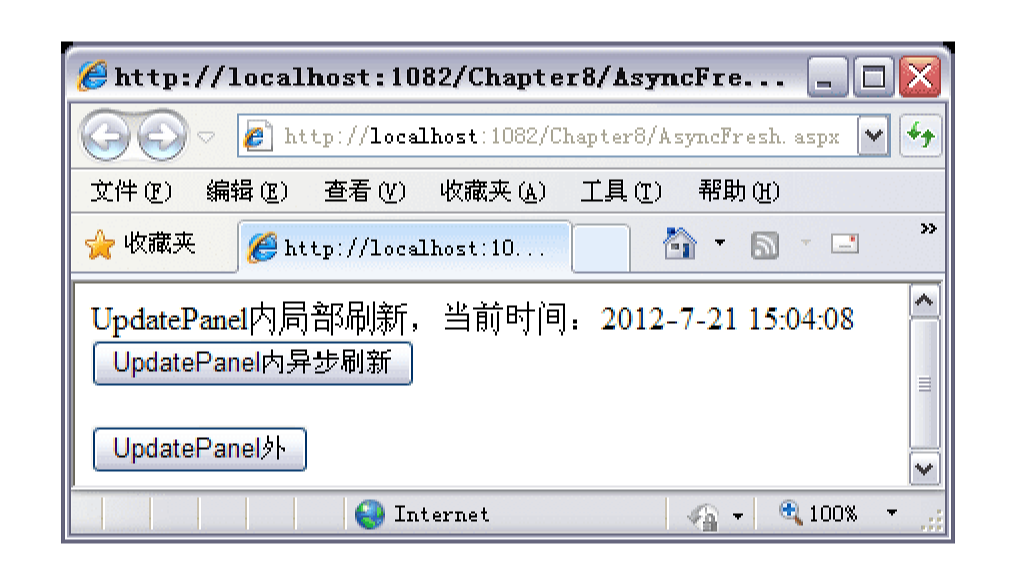


图 8-3　页面运行效果图

可以看到，虽然单击每个按钮都能获取最新时间并显示，但页面刷新效果却不同。

4. 使用 Triggers 属性

在 UpdatePanel 中有两种 Triggers，分别为 AsyncPostBackTrigger 和 PostBackTrigger。AsyncPostBackTrigger 用来指定某个服务器控件作为该 UpdatePanel 的异步更新触发器，PostBackTrigger 用来指定在 UpdatePanel 中的某个服务器控件作为同步更新触发器。异步更新触发器所引发的回传叫异步回传，引发页面局部更新。而同步更新触发器所引发的回传叫同步回传，引发传统的整页回传。

例 8-2：利用 Triggers 属性指定服务器控件的回传方式。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 UseTrigger.aspx 的页面。

(3) 切换到页面的“设计”视图，添加一个 ScriptManager 和一个 UpdatePanel 控件。

(4) 在 UpdatePanel 内部添加一个 Label 控件和两个 Button 控件，同时在 UpdatePanel 外面也添加两个 Button 控件。

(5) 设置按钮的 Text 属性分别为“UpdatePanel 内完整页面刷新”、“UpdatePanel 内局部刷新”、“UpdatePanel 外”和“UpdatePanel 外异步刷新”。控件的布局如图 8-4 所示。

(6) 选中 UpdatePanel 控件，通过“属性”面板设置 Triggers 属性，这是一个集合属性，单击属性右侧的按钮，打开“UpdatePanelTrigger 集合编辑器”对话框，单击“添加”按钮右侧的倒三角形，打开一个下拉菜单，如图 8-5 所示，可以添加两类元素：PostBackTrigger 和 AsyncPostBackTrigger。本例中设置 Button1 为 PostBackTrigger，Button4 为 AsyncPostBackTrigger。

技巧：

如果“属性”面板中找不到 Triggers 属性，则可能是页面中没有添加 ScriptManager 控件。

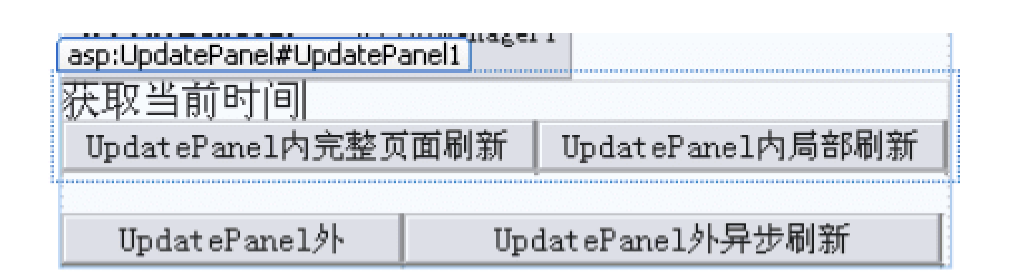


图 8-4　控件的页面布局

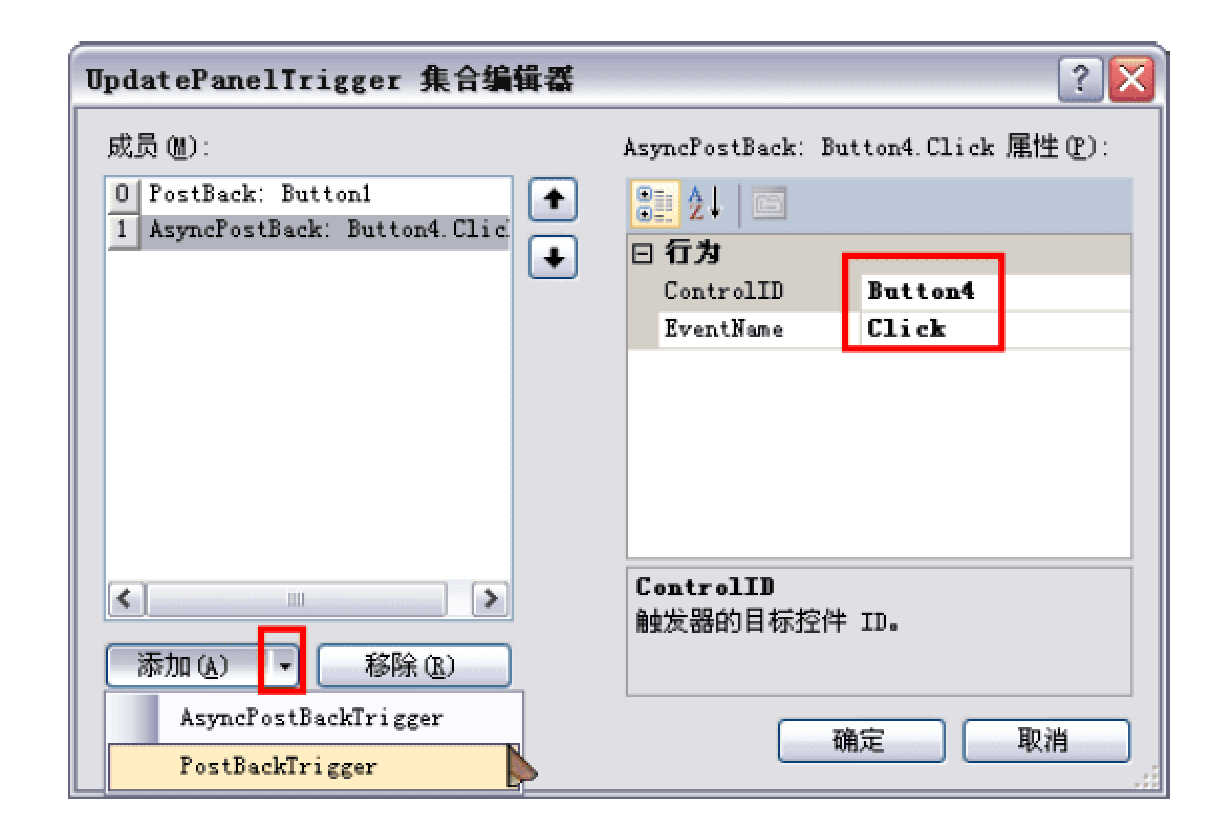


图 8-5　“UpdatePanelTrigger 集合编辑器”对话框

(7) 分别为 4 个 Button 控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 内整个页面刷新，当前时间：" + DateTime.Now.ToString();
}
protected void Button2_Click(object sender, EventArgs e)
{
    Label1.Text = "局部刷新无闪烁，当前时间：" + DateTime.Now.ToString();
}
protected void Button3_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 外整个页面刷新，当前时间：" + DateTime.Now.ToString();
}
protected void Button4_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 外异步刷新无闪烁，当前时间：" + DateTime.Now.ToString();
}
```

(8) 编译并运行程序，在浏览器中打开 UseTrigger.aspx 页面。分别单击不同的按钮，观察有什么不同，可以看到，使用了 Triggers 属性后，虽然 Button1 按钮在 UpdatePanel 内部，但实现的却是整个页面的更新，而在 UpdatePanel 外面的 Button4 按钮却实现了局部刷新的效果，如图 8-6 所示是单击 Button4 后局部刷新无闪烁的情况。

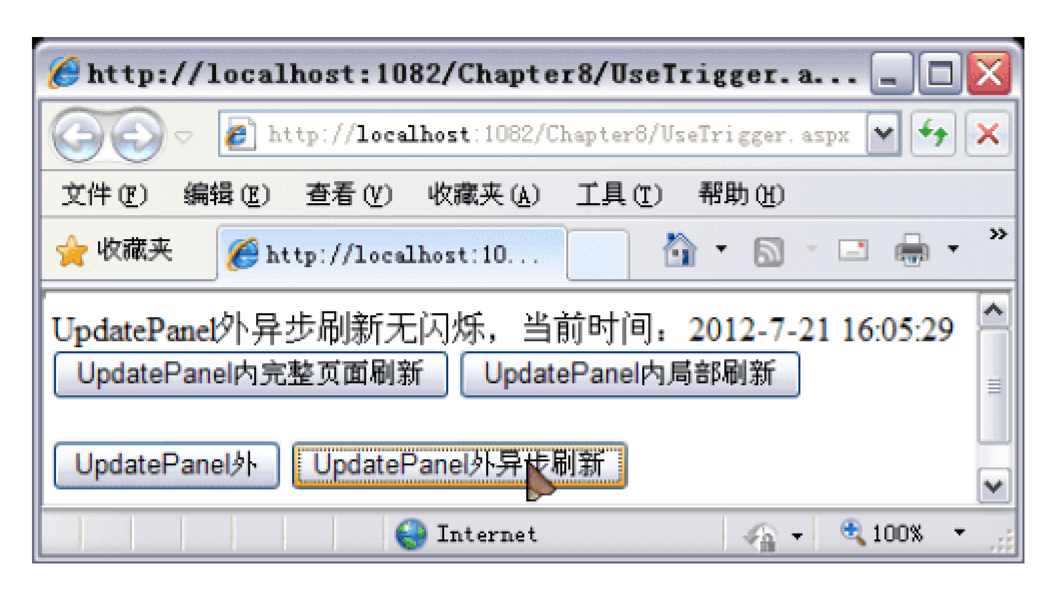


图 8-6　UpdatePanel 外按钮的局部刷新

技巧：

默认情况下，在 UpdatePanel 内部的服务器控件采用的是异步回传方式，实现 UpdatePanel 的局部更新；而在 UpdatePanel 外部的服务器控件采用的是同步回传方式，实现整个页面的刷新。而通过 Triggers 属性可以将 UpdatePanel 外的控件设置为异步刷新，所以在使用母版页的时候，可以在母版页中放置 ScriptManager 控件，在内容页使用 UpdatePanel，并通过 Triggers 属性设置母版页中的控件实现局部更新

5. 在同一页面上使用多个 UpdatePanel

ASP.NET 并没有限制在一个页面中使用多少个 UpdatePanel，所以可以为不同的区域添加不同的 UpdatePanel 控件。由于 UpdatePanel 默认的 UpdateMode 是 Always，所以，如果页面上有一个局部更新被触发，则所有的 UpdatePanel 都将更新，要想只更新某个 UpdatePanel，只需把 UpdateMode 属性设置为 Conditional 即可。

下面的例子就包括两个 UpdatePanel 控件，其中一个用来输入数据，而另一个则用来显示数据，两个 UpdatePanel 的 UpdateMode 属性都设置为 Conditional，当单击“添加”按钮时，两个 UpdatePanel 都更新，单击“取消”按钮时，只有 UpdatePanel2 更新。

例 8-3：同一页面中使用多个 UpdatePanel。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 MultiUpdatePanel.aspx 的页面。

(3) 切换到页面的“设计”视图，添加一个 ScriptManager 和两个 UpdatePanel 控件。

(4) 在 UpdatePanel1 内部添加 ListBox 控件和一个 Label 控件，ListBox 控件中的选项通过 UpdatePanel2 中的 TextBox 控件添加，Label 控件用于显示当前时间。

(5) 在 UpdatePanel2 中添加一个 Label 控件、一个 TextBox 控件和两个 Button 控件。其中，Label 控件还是用于显示当前时间，TextBox 控件用于输入要添加到 ListBox 中的选项值，单击“添加”按钮将添加项到 ListBox 中，单击“取消”按钮将清空 TextBox 控件。

(6) 设置两个 UpdatePanel 控件的 UpdateMode 属性为 Conditional。设置 UpdatePanel1 的 Triggers 属性为 Button1 异步刷新 AsyncPostBackTrigger。

(7) 切换到“源”视图，修改相应的代码如下所示：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
        <ContentTemplate>
            <fieldset style="width:180px;">
            <legend >UpdatePanel1</legend>
            <asp:ListBox ID="ListBox1" runat="server" Width="108px"></asp:ListBox>
            <br />
            <asp:Label ID="Label1" runat="server" ><%=DateTime.Now %></asp:Label>
        </ContentTemplate>
        <Triggers>
```

```
                <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
            </Triggers>
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <fieldset style="width:180px;">
                <legend >UpdatePanel2</legend>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <br />
                <asp:Button ID="Button1" runat="server" Text="添加" onclick="Button1_Click" />
                <asp:Button ID="Button2" runat="server" Text="取消" onclick="Button2_Click" />
                <br />
                <asp:Label ID="Label2" runat="server" ><%=DateTime.Now %></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
```

(8) 添加两个 Button 控件的事件处理程序，代码如下：

```
 protected void Button1_Click(object sender, EventArgs e)
{
    ListBox1.Items.Add(TextBox1.Text);
    TextBox1.Text = "";
}
protected void Button2_Click(object sender, EventArgs e)
{
    TextBox1.Text = "";
}
```

(9) 编译并运行程序，在浏览器中打开 MultiUpdatePanel.aspx 页面，在 UpdatePanel2 中的 TextBox 控件中输入一个值，单击“添加”按钮，该值将添加到 UpdatePanel1 中的 ListBox 中，同时，两个显示时间的 Label 控件都进行了更新，如图 8-7 所示。

(10) 单击“取消”按钮，将只更新 UpdatePanel2，如图 8-8 所示，只有 UpdatePanel2 中的时间进行了更新。




图 8-7　同时更新多个 UpdatePanel 控件

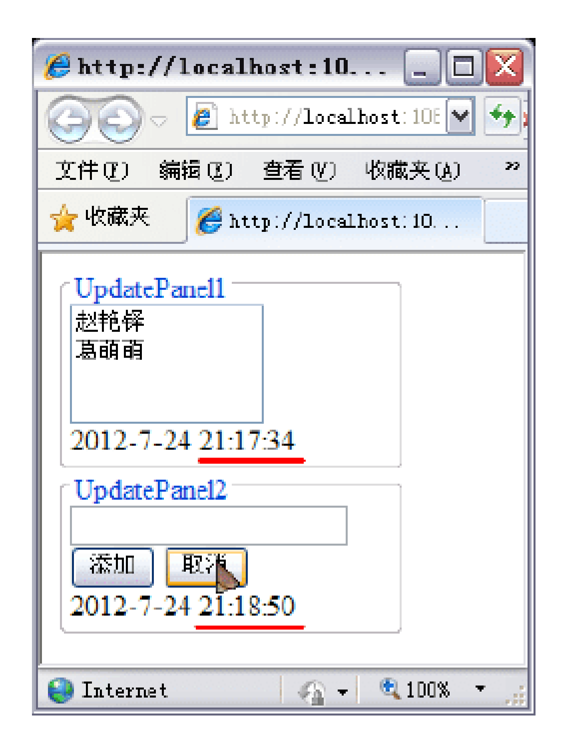


图 8-8　只更新指定的 UpdatePanel 控件

8.2.3　UpdateProgress 控件

虽然使用 UpdatePanel 和 ScriptManager 已经足以创建无闪烁页面，但 ASP.NET AJAX 提供了更多控件来增强用户在启用了 Ajax 的 Web 站点中的体验。改进用户体验的方法之一是使用 UpdateProgress 控件，另一种则是使用 Timer 控件。

UpdateProgress 控件一般与 UpdatePanel 控件联合使用，即在 UpdatePanel 异步更新过程中，显示提示信息。这些信息可以是一段文字、一个进度条或某个动画。当异步更新完成时，提示信息自动消失。

1. UpdateProgress 控件的属性

UpdateProgress 控件的常用属性如表 8-3 所示。

表 8-3　UpdateProgress 控件的常用属性

属　性	说　　明
AssociateUpdatePanelID	设置哪个 UpdatePanel 控件产生的回送会显示 UpdateProgress 的内容，当关联的 UpdatePanel 控件忙于刷新时，就会显示在<ProgressTemplate>元素中定义的内容。通常要在模板中放入文本或动画图像(也接受其他标记)来让用户知道正在发生的事情
DisplayAfter	当引发回送后多少毫秒会显示 UpdateProgress 控件的内容，默认值是 500 毫秒
DynamicLayout	设置 UpdateProgress 控件的显示方式。如果为 true(默认值)，当 UpdateProgress 控件不显示的时候不占用空间；为 false，当 UpdateProgress 控件不显示的时候仍然占用空间
ProgressTemplate	获取或设置定义 UpdateProgress 控件内存的模板

如果没有设定 UpdateProgress 控件的 AssociateUpdatePanelID 属性，则任何一个异步更新都会使 UpdateProgress 控件显示出来。相反，如果将 UpdateProgress 控件的 AssociateUpdatePanelID 属性设置为某个 UpdatePanel 控件的 ID，那么只有该 UpdatePanel 控件引发的异步更新才会使相关联的 UpdateProgress 控件显示出来。

注意：

必须为 UpdateProgress 控件定义模板，否则，在 UpdateProgress 控件的 Init 事件中会抛出异常。可以通过将标记添加到 ProgressTemplate 元素，以声明的方式指定 ProgressTemplate 属性。如果要动态创建 UpdateProgress 控件，则应在页面的 PreRender 事件中或之前进行创建。

2. 使用 UpdateProgress 控件

例 8-4：使用 UpdateProgress 控件给用户提供反馈信息。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 UpdateProgress.aspx 的页面。

(3) 在“解决方案资源管理器”窗口中，在项目根目录下新建一个 images 文件夹，然后添加进度条动画文件 progress.gif 到该文件夹中。

(4) 切换到页面的“设计”视图，添加一个 ScriptManager、一个 UpdatePanel 控件和一个 UpdateProgress 控件。

(5) 在 UpdatePanel 控件中添加一个 Label 控件和一个 Button 控件。其中，Label 控件用于显示当前时间，设置 Button 控件的 Text 属性为“提交”。

(6) 在 UpdateProgress 控件中添加文本“局部刷新进行中，请稍候…”和一个 Image 控件，Image 控件的 ImageUrl 属性指向前面的进度条动画文件 progress.gif。

(7) 添加按钮控件的单击事件处理程序，为了演示进度条功能，在程序中等待 3 秒钟再刷新页面，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);   //等待 3 秒
    Label1.Text = DateTime.Now.ToString();
}
```

(8) 编译并运行程序，运行效果如图 8-9 所示。

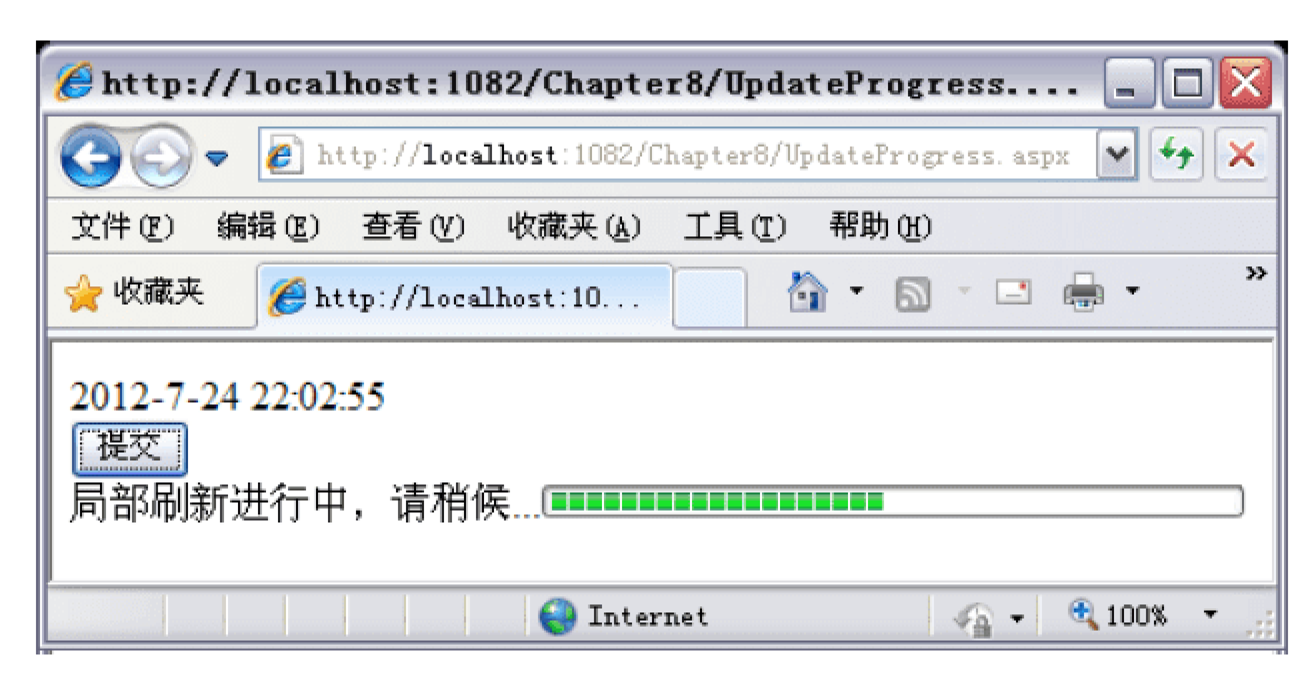


图 8-9　使用进度提示控件

本例中只有一个 UpdateProgress 控件，但在实际操作中也可以在一个页面中使用多个 UpdateProgress 控件，通过设置 AssociateUpdatePanelID 属性来指点相关联的 UpdatePanel 控件。

3. 进度条的取消功能

UpdateProgress 控件还支持另一个技术细节，即支持取消命令按钮。当用户单击了“取消”按钮时，异步回调将立即被终止，该 UpdateProgress 控件将消失，页面恢复到原来的状态。

例 8-5：为进度条增加“取消”命令按钮，当用户单击了“取消”按钮时，结束异步回调，恢复到原来的状态。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 CancelUpdateProgress.aspx 的页面。

(3) 切换到页面的“设计”视图，添加一个 ScriptManager、一个 UpdatePanel 控件和一个 UpdateProgress 控件。

(4) 在 UpdatePanel 控件中添加一个 Label 控件和一个 Button 控件，Label 控件用于显示当前时间，设置 Button 控件的 Text 属性为“提交”。

(5) 在 UpdateProgress 控件中添加文本“局部刷新进行中，请稍候…”和一个 Image 控件，Image 控件的 ImageUrl 属性指向前面的进度条动画文件 progress.gif，然后，继续添加一个 HTML 控件 Input(Button)，设置按钮的 Text 属性为“取消”。

(6) 切换到“源”视图，在上述所有控件的下面添加如下客户端 JavaScript 代码块：

```
<script type="text/javascript">
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    $addHandler($get('Button2'), 'click', AbortPostBack);
    prm.add_initializeRequest(InitializeRequest);
    function InitializeRequest(sender, args) {
        if (prm.get_isInAsyncPostBack())
            args.set_cancel(true);
    }
    function AbortPostBack() {
        if (prm.get_isInAsyncPostBack())
            prm.abortPostBack();
    }
</script>
```

上述代码将“取消”按钮的 click 事件连接到 AbortPostBack()方法，该方法用来取消当前的回发请求。

(7) 添加“提交”按钮控件的单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);    //等待 3 秒
    Label1.Text = DateTime.Now.ToString();
}
```

(8) 编译并运行程序，单击“提交”按钮，在回发过程中单击“取消”按钮，将取消回发，页面恢复到原来的状态，如图 8-10 所示。

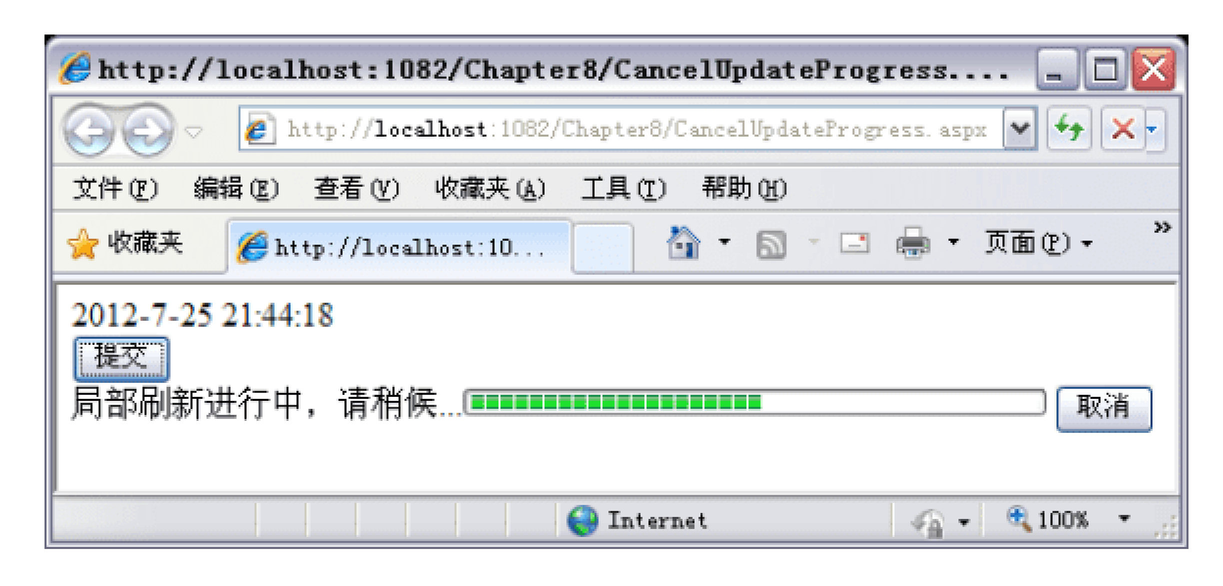




图 8-10　取消异步回发

注意：

请不要将客户端方法与服务器端的事件处理相混淆：客户端方法允许浏览器捕获相应的事件，并使用 JavaScript 代码进行处理。这一过程根本不涉及到服务器端。事实上，当用户取消一个操作时，服务器端仍然或继续处理该请求，只是浏览器此时已关闭连接并停止监听。

8.2.4　Timer 控件

Timer 控件是 ASP.NET AJAX 中又一个重要的服务器控件。通过它可以完成局部页面的定时更新，从而实现局部页面定时刷新、图片自动播放以及超时自动退出等功能。

1. 属性和事件

Timer 控件常用属性和事件如表 8-4 所示。

表 8-4　Timer 控件的常用属性和事件

属性和事件	说　　明
Interval	该属性用于指定间隔时间，其设置值的单位是毫秒，默认值则是 60000 毫秒
Enabled	该属性用于表示是否允许 tick 事件
Tick	该事件在 Interval 指定的间隔到期后触发

需要注意的是，将 Timer 控件的 Interval 属性设置为较小的值会使得回送频率增加，也很容易使 Web 服务器的流量大增，对整体资源耗用与效率都会造成不良的影响。因此应尽量在确实需要的时候才使用 Timer 控件来定时更新页面上的内容。

Timer 控件在 UpdatePanel 控件的内外是有区别的。当 Timer 控件在 UpdatePanel 控件内部时，JavaScript 计时组件只有在一次回传完成后才会重新建立。也就是说，直到网页回传完成之前，定时器间隔时间不会从头计算。例如，用户设置 Timer 控件的 Interval 属性值为 3000ms(3 秒)，但是回传操作本身却花了 2 秒才完成，那么，下一次的回传将发生在前一次回传被引发之后的 5 秒。而如果 Timer 控件位于 UpdatePanel 控件之外，当定时器间隔到期以后，定时器间隔时间会立刻重新计算，下一次回传将发生在前一次回传被引发之后的 3 秒。

2. 使用 Timer 控件定时更新 UpdatePanel

Timer 控件的用法非常简单。该控件按照指定的时间间隔激活其 Tick 事件。在这个事件处理程序中添加要刷新页面的代码即可。

例 8-6：在 UpdatePanel 内部使用 Timer 控件。实现图片的自动刷新。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 TimerTest.aspx 的页面。

(3) 向网站根目录下的 images 文件夹中添加需要循环显示的图片文件 Hearts1.gif、

Hearts2.gif…Hearts13.gif。

(4) 切换到 TimerTest.aspx 的“设计”视图，添加一个 ScriptManager 和一个 UpdatePanel 控件。

(5) 在 UpdatePanel 控件中添加一个 Label 控件、一个 Timer 控件和一个 Image 控件，设置 Timer 控件的 Interval 属性为 3000，设置 Image 控件的 ImageUrl 属性为 images 目录下的 Hearts1.gif。

(6) 对应“源”视图中生成的代码如下：

```
<h3>使用 Timer 控件循环显示红桃 1-K</h3>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Timer ID="Timer1" runat="server" Interval="3000"
ontick="Timer1_Tick">
        </asp:Timer>
        <asp:Label ID="Label1" runat="server" Text="显示当前时间"></asp:Label>
        <asp:Image ID="Image1" runat="server" ImageUrl="~/images/Hearts1.gif" />
    </ContentTemplate>
</asp:UpdatePanel>
```

(7) 添加 Timer 控件的 Tick 事件处理程序和页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack )
    {
        ViewState["count"] = 1;//设置网页上的变量
    }
}
protected void Timer1_Tick(object sender, EventArgs e)
{
    ViewState["count"] = (int)ViewState["count"] % 13 + 1;
    System.Threading.Thread.Sleep(2000);//等待 2 秒
    Label1.Text = DateTime.Now.ToString();
    Image1.ImageUrl = string.Format("~/images/Hearts{0}.gif", ViewState["count"]);
}
```

上述代码中使用到了已经学习过的视图状态来存放一个计数变量。

(8) 编译并运行程序，由于 Timer 控件在 UpdatePanel 控件内部，Interval 属性为 3 秒，程序内部等待 2 秒，所以看到的结果是每 5 秒自动刷新一次时间，同时显示下一张扑克牌，循环显示红桃 1-K，如图 8-11 所示。





图 8-11　使用 Timer 控件定时刷新页面

3. 定时更新多个 UpdatePanel

使用 Timer 控件定时更新多个 UpdatePanel 控件，可以使用一个 Timer 控件，也可以使用多个 Timer 控件。

例 8-7：使用两个 Timer 控件定时更新两个 UpdatePanel 控件，Timer 控件均被放在 UpdatePanel 控件的外面，并将 Timer 控件配置为 UpdatePanel 控件的触发器。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 TimerTest2.aspx 的页面。

(3) 切换到 TimerTest.aspx 的“设计”视图，添加一个 ScriptManager 和两个 UpdatePanel 控件。在两个 UpdatePanel 控件中各添加一个 Label 控件，用于显示当前时间。

(4) 在两个 UpdatePanel 控件之外，添加两个 Timer 控件，设置 Timer1 的 Interval 属性为 2000，即每隔 2 秒更新一次，设置 Timer2 的 Interval 属性为“3000”，即每隔 3 秒更新一次。

(5) 选中 UpdatePanel1 控件，通过“属性”面板设置其 Triggers 属性，在打开的“UpdatePanelTrigger 集合编辑器”对话框中添加一个 AsyncPostBackTrigger，设置 ControlID 为 Timer1，EventName 为 Tick。

(6) 同样地，设置 UpdatePanel2 的 AsyncPostBackTigger 为 Timer2。

(7) 对应“源”视图中生成的代码如下：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
        </Triggers>
    </asp:UpdatePanel>
    <asp:UpdatePanel ID="UpdatePanel2" runat="server">
```

```
                <ContentTemplate>
                    <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
                </ContentTemplate>
                <Triggers>
                    <asp:AsyncPostBackTrigger ControlID="Timer2" EventName="Tick" />
                </Triggers>
            </asp:UpdatePanel>
            <asp:Timer ID="Timer1" runat="server" Interval="2000" ontick="Timer1_Tick">
            </asp:Timer>
            <asp:Timer ID="Timer2" runat="server" Interval="3000" ontick="Timer2_Tick">
            </asp:Timer>
        </div>
```

(8) 添加 Timer 控件的 Tick 事件处理程序，代码如下：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel1  更新于：  " + DateTime.Now.ToString();
}
protected void Timer2_Tick(object sender, EventArgs e)
{
    Label2.Text = "UpdatePanel2  更新于：  " + DateTime.Now.ToString();
}
```

(9) 编译并运行程序，可以看到，UpdatePanel1 控件每 3 秒更新一次，UpdatePanel2 控件每 2 秒更新一次，如图 8-12 所示。

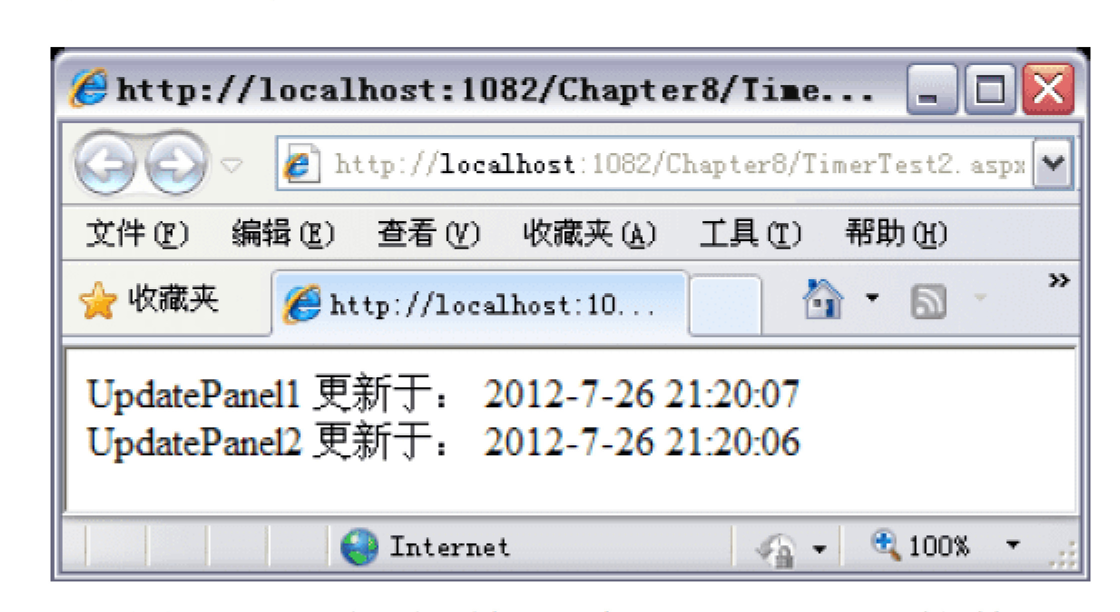


图 8-12　定时刷新 2 个 UpdatePanel 控件

8.2.5　ScriptManagerProxy 控件

ScriptManagerProxy 控件是内容页面与母版页中定义的 ScriptManager 控件之间的桥梁。在页面中，ScriptManagerProxy 控件的外观和操作与标准控件 ScriptManager 很相似。但是，ScriptManagerProxy 控件实际上只是一个 proxy 类，且该类可以将其所有的设置传递给母版页中真正的 ScriptManager 控件。

本书第 9 章将介绍 Web 服务的时候会用到该控件，此处不过多介绍。

8.3　客户端 ASP.NET AJAX Library

客户端 ASP.NET AJAX Library 的功能非常强大，并且能够提供浏览器中纯 JavaScript 所缺少的许多功能。Microsoft 从.NET Framework 中吸取了许多好的功能，然后将它们转换为客户端 JavaScript 对应的功能。这就意味着可以在支持 JavaScript 的 Web 站点中使用熟悉的.NET 概念，即使 JavaScript 不支持这些概念。

客户端库包括 6 个顶级的命名空间(包括根命名空间 Sys)和 1 个全局命名空间，它们允许访问 30 多个类，而这些类具有数百种有用的方法，可用来帮助创建富客户端 Web 界面。

例如，可以使用客户端框架建立和处理页面事件(如 load 和 unload)，使用 WebRequest 类对其他的 Web 页面发出请求，根据不同的客户端设置来呈现不同格式的数据，执行数据绑定(绑定的数据源有多种，如可以将 Web 服务绑定到用户的界面控件中，从而显示并编辑它们)以及其他情况。

全局命名空间中包含了一些成员和类型，它们扩展了 JavaScript 原先设计的特性。还包含了在 JavaScript 中发现的类型，如 Array、Boolean、Error 、Number、 Object 和 String，这些类型已经被扩展为包含模仿.NET Framework 的行为。如表 8-5 所示列出了这些类型的一些常用的方法。

表 8-5　全局命名空间中的成员和类型的常用方法

类　　型	方　　法	用　　途
String	format	使用指定的格式字符串和参数返回一个格式化的字符串，例如： var str = String.format('Hello {0}', name);
	endsWith startsWith	用于确定一个字符串是否以另一个字符串作为开始或结束，例如： var isGemm = 'Ge Mengmeng'.startsWith('Ge');
	trim trimEnd trimStart	删除字符串前面和后面的空格，例如： var trimmed = ' Zhao Yanduo '.trim();// results is Zhao Yanduo
Boolean	parse	将保留包含有文本 true or 或 false 的字符串转换成真正的布尔值。出现其他值时则抛出一个异常，例如： var isTrue = Boolean.parse('true');
Date	format	可以获得日期的不同的格式化字符串表示，例如： alert(new Date().format('f'));

客户端 ASP.NET AJAX Library 很大，所以并不是总能很容易地找到需要的方法、类或命名空间。为了更有效地使用客户端库，可以使用一些很好的资源。而首选就是 VWD 提供的“智能提示”，它可以在不同的类型中提供可用成员。

例如，如图 8-13 所示，VWD 推测变量 name 的类型是字符串，这是因为该变量提供了与字符串相关的方法，如 substring 和 trim，这两个方法已经被客户端库添加到 String 类型中了。如果指定一个数字给变量，就会得到一个与数字类型对应的列表，如图 8-14 所示。

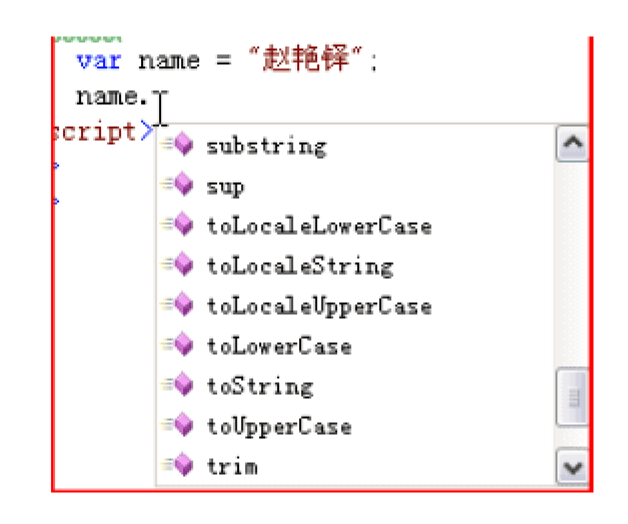


图 8-13　智能提示推测变量为字符串

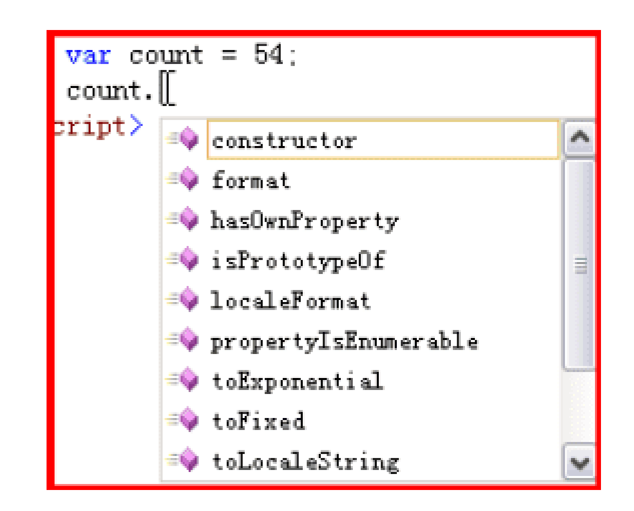


图 8-14　数字类型变量对应的方法列表

许多成员已经被文档化，因此可以得到一个很好的提示文本来说明如何使用这些成员。如果没有看到正确的方法出现，则可能是页面(或母版页)中没有 ScriptManager 控件。而没有该控件，客户端 JavaScript 框架将无法使用，因此 VWD 就会禁止使用浏览器的最终页面中不可用的功能。有时“智能提示”很可能没有在列表中显示所期望的成员，但是这并不能说明这些方法不可用。因为 JavaScript 不是一种强类型的语言，所以 VWD 需要做许多的解析、推断和猜测工作来提供正确的选项。

例 8-8：客户端 ASP.NET AJAX Library 的一个简单应用。

(1) 启动 VWD 2010，打开网站 Chapter8。

(2) 通过“添加新项”对话框添加一个名为 AjaxLibrary.aspx 的页面。

(3) 切换到 AjaxLibrary.aspx 的“设计”视图，添加一个 ScriptManager 控件。

(4) 在<ScriptManager>的结束标记下方，添加一个 Input (Text)和一个 Input (Button)控件，方法是从工具箱的 HTML 类别中拖动它们。通过使用纯 HTML 元素而不是 ASP.NET 服务器控件，可以看到要写的代码在客户端执行。将按钮的 value 设置为“提交”。相应的代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
姓名：<input id="TextName" type="text" />
<input id="Button1" type="button" value="提交" />
```

(5) 在这两个 HTML 控件的下面，添加如下客户端 JavaScript 代码块：

```
<script type="text/javascript">
    $addHandler($get('Button1'), 'click', Hello);
    function Hello() {
        var name = $get('TextName').value;
        if (name.length == 0)
            alert("请输入姓名");
        else
            alert("Hello " + name);
    }
</script>
```

其中，$addHandler 是 Ajax 框架中定义的 Sys.UI.DomEvent 类的 addHandler 方法的一

个快捷方式。可以用它在页面中注册这些对象特定事件的事件处理程序。这与在 C#服务器端代码中看到的事件处理程序类似。$get 用来获得对 HTML 标记的引用。

技巧：

$get 和$addHandler 最大的好处是可以在任何站点中使用它们。所需做的就是在母版页或内容页内包含 ScriptManager 控件，并准备使用客户端框架。

(6) 编译并运行程序，在文本框中输入一个名字，单击“提交”按钮，将弹出相应的欢迎对话框，如图 8-15 所示。

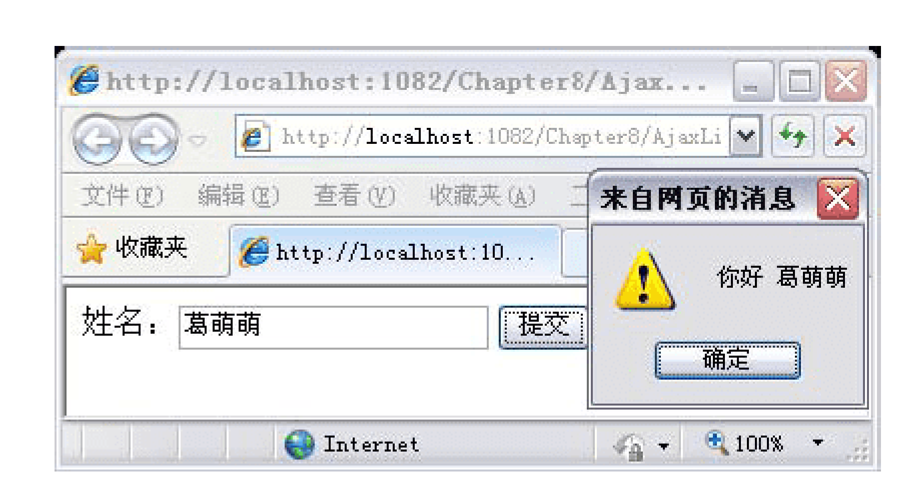


图 8-15　客户端 AJAX Library 应用示例

说明：

ASP.NET AJAX 包含的内容还有很多，无法在此都一一介绍。ASP.NET AJAX 的服务器和客户机端部分可能是最大的、使用最多的功能，其他的功能也都是比较实用的，例如，ASP.NET AJAX 控件工具箱，它是一个非常好的扩展控件工具包，带有诸如日历扩展器和能自动完成的文本框这样的功能。包括四十多个免费的扩展控件，而且一直都在增加，可以在网站 http://www.asp.net/ajax/AjaxControlToolkit/Samples 上查看和下载控件工具箱。

8.4　本章小结

Ajax 应用广泛，可以给站点增加许多功能。它可以分为两个不同的部分，服务器端控件和客户端 JavaScript Framework。要使用 Ajax 功能增强 Web 站点，可以选择不同的 Ajax 架构。Microsoft ASP.NET AJAX 除了提供实现无闪烁页面的控件之外，还提供了更多的服务器控件来创建富交互式的且有响应的用户界面。本章重点介绍了 ASP.NET AJAX 控件的具体用法，以及客户端 AJAX Library。通过本章的学习，读者应掌握快速创建无闪烁页面的方法。

8.5　思考和练习

1. 工具箱中的 AJAX Extensions 类别中的 ScriptManager 和 ScriptManagerProxy 控件有何区别？

2. UpdatePanel 控件有什么作用？如何让 UpdatePanel 控件外部的按钮进行异步刷新？

3. 如何让用户知道部分页面更新正在进行？

4. UpdateProgress 控件的 AssociatedUpdatePanelID 属性有什么用？

5. $addHandler 有什么作用？

6. 上机操作。

- 新建一个网站，在 Default.aspx 页面中的左上角显示当前的时间，要求采用局部刷新技术。
- 添加一个网页，实现相册功能，通过“上一个”和“下一个”按钮，局部刷新 Image 控件，显示不同的图片。

第9章　Web服务

Web 服务使得 Internet 成为一个可以无限扩展、拥有无限潜力的分布式计算平台。任何设备可以随时随地访问 Internet 上的 Web 服务。Web 服务具有基于组件的开发和 Web 开发两者的优点。本章将介绍 ASP.NET Web 服务体系，使用 VWD 创建和调用 Web 服务的机制，以及在 AJAX 站点中使用 Web 服务和 Web 方法。

本章学习目标：

- Web 服务的工作原理
- 创建 Web 服务
- 调用 Web 服务
- 支持 AJAX 的 Web 服务
- 在 AJAX 站点中调用页面方法

9.1　Web 服务入门

Web 服务奠定了下一代 Web 应用程序的基础。无论客户应用程序是 Windows 应用程序，还是 ASP.NET Web 应用程序，无论客户程序运行在 Windows、Pocket Windows 或其他操作系统上，它们都可以通过 Internet 使用 Web 服务定期通信。Web 服务是服务器端的程序，用来监听来自客户应用程序的消息，并返回特定的信息。这些信息可能来自 Web 服务本身，也可能是同一个域中的其他组件，或其他 Web 服务。

9.1.1　Web 服务概述

简单地讲，Web 服务是一个基于因特网的可通过 Web 被远程调用的应用程序模块(API)，例如网站中如果想提供天气预报的服务，不用自己实现天气预报的功能，只需调用其他公司提供的免费或付费 Web 服务即可。

- 服务就是一个软件，它和客户端应用程序没有很紧密地耦合或关联。服务是可以被动态地发现及组合成其他软件的软件实体。
- Web 服务是一种基于 XML、JSON、SOAP、HTTP、UDDI 和 WSDL 等一系列标准实现的分布式计算技术和软件组件。
- Web 服务提供了一个松耦合和跨平台的分布式计算环境，它是一个与操作系统、程序设计语言、机器类型以及运行环境都无关的平台，能够实现网络上应用的共享，并可用于复杂的系统集成。

微软为 Web 服务下的定义是通过标准的 Web 协议可编程访问的 Web 组件。每个 Web 服务的实现是完全独立的。Web 服务具有基于组件的开发和 Web 开发两者的优点，是 Microsoft 的.NET 程序设计模式的核心。

国际标准化组织 W3C 为 Web 服务下的定义是一个通过 URL 识别的软件应用程序，其界面及绑定能用 XML 文档来定义、描述和发现，使用基于 Internet 协议上的消息传递方式与其他应用程序进行直接交互。

1. Web 服务的影响

(1) Web 服务支持在 Web 站点上放置可编程的元素，用户可以抓取已有的元素，构成自己的新服务。

(2) 能进行基于 Web 的分布式计算和处理，能很好地兼容现有的 Web 技术。

(3) Web 服务使得 Internet 成为一个可以无限扩展、拥有无限潜力的分布式计算平台。

(4) 任何设备都可以随时随地访问 Internet 上的 Web 服务。

(5) 软件模块充分复用、计算机资源充分共享、信息无缝共享和交流。

(6) 利用 Web 服务，公司和个人将能够迅速且廉价地向整个国际互联网提供他们的服务，进而建立全球范围的联系，在广泛的范围内寻找可能的合作伙伴。

2. Web 服务的主要特征

(1) 互操作性：一个 Web 服务可以与其他 Web 服务交互，协同工作；可以使用任何语言开发 Web 服务或使用他人提供的 Web 服务；开发环境可以异构。

(2) 普遍性：Web 服务使用 HTTP 和 XML 进行通信，支持这些技术的设备都可以拥有和访问 Web 服务。

(3) 松散耦合：Web 服务的实现对使用者透明，当服务的实现发生变动时不影响用户使用。

(4) 高度可集成能力：Web 服务和采用了简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范，屏蔽了平台的异构性，CORBA、DCOM 和 EJB 等都可通过它进行互操作。

9.1.2　ASP.NET Web 服务体系

.NET 平台和 ASP.NET 在创建和使用 Web 服务方面提供了广泛的支持。这些技术赋予用户一个优秀的、简单易用的平台，从而可以快速有效地创建和使用 Web 服务。如图 9-1 所示是 ASP.NET Web 服务的体系结构。

ASP.NET Web 服务体系包括客户端应用程序、ASP.NET Web 服务程序以及一些文件，如：代码文件、.asmx 文件和编译后的.dll 文件。还包括一台 Web 服务器来存载 Web 服务程序和客户端。如果需要，还可以有一台数据库服务器来存取 Web 服务中的数据。

XML 或 JSON 是数据的格式，SOAP 是调用 Web 服务的协议，WSDL 是描述 Web 服务的格式，而 UDDI 是 Web 服务发布、查找和利用的组合。

- SOAP(Simple Object Access Protocol)：SOAP 是一套用于 Web 服务端和客户端通信的标准消息控制协议，要调用 Web 服务上的一个方法，该调用必须转换为 SOAP 消息，因为它是在 WSDL 文档中定义的。如图 9-2 所示是 SOAP 消息的一部分。SOAP 封装(envelop)把所有的 SOAP 消息封装在一个块中。SOAP 封装本身由两部分组成：SOAP 标题和 SOAP 体。标题是可选的，它定义了客户机和服务器应如何处理 SOAP 体。SOAP 体是必须有的，它包括发送的数据，通常 SOAP 体中的信息是要调用的方法和序列化的参数值。SOAP 服务器在 SOAP 消息的消息体中返回值。

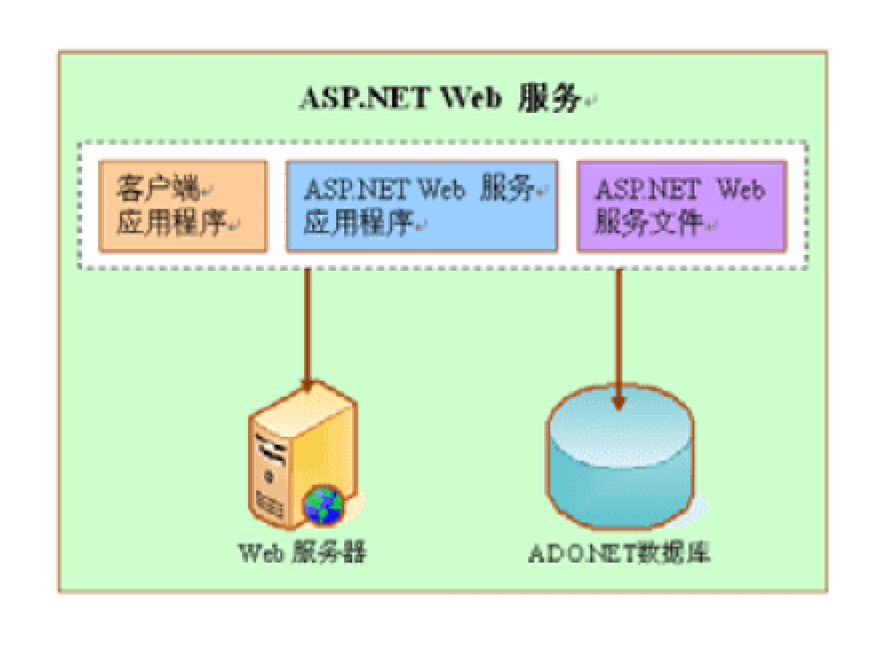


图 9-1　ASP.NET Web 服务体系

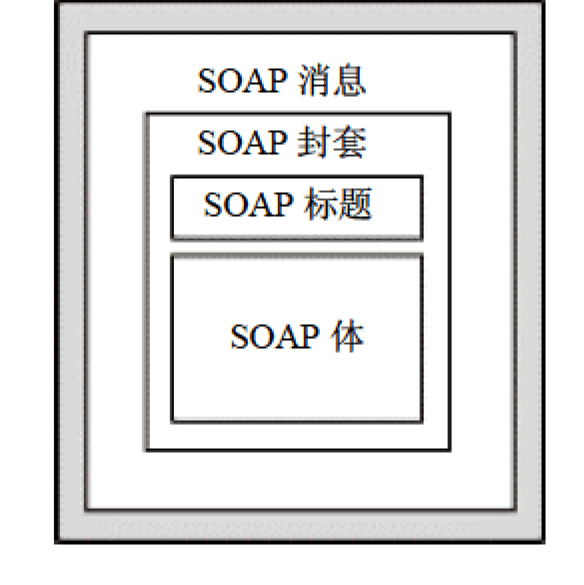


图 9-2　SOAP 消息

- WSDL(Web Services Description Language)：WSDL 是 Web 服务描述语言。可以认为 WSDL 文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。换句话说，WSDL 对于 SOAP 的作用就像 IDL 对于 CORBA 或 COM 的作用。通常 WSDL 文档是由软件生成和使用的。
- UDDI(Universal Description Discovery and Integration)：UDDI 是 Web 服务的黄页。与传统黄页一样，用户可以搜索提供所需服务的公司，阅读了解所提供的服务，然后与某人联系以获得更多信息。当然用户也可以提供 Web 服务而不在 UDDI 中注册，就像在地下室开展业务，依靠的是口头吆喝；但是如果希望拓展市场，则需要 UDDI 以便能被客户发现。

9.1.3　支持 AJAX 的 Web 服务

ASP.NET AJAX 提供了完整的架构以从客户端 JavaScript 调用 ASP.NET Web 服务。设计者可以轻松地用 AJAX 把服务器端的数据和功能集成到用户响应的 Web 页面中，而所需要做的就是用 ScriptService 属性来标识 Web 服务。ASP.NET AJAX 框架会为 Web 服务自动生成 JavaScript 代理，然后通过使用代理来调用 Web 方法。

JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，易于阅读和编写，并且易于机器解析和生成。JSON 采用完全独立于语言的文本格式，使用了类似于 C 语言家族的习惯。这些特性使 JSON 成为理想的数据交换语言。

JSON 建构于以下两种结构。

(1) “名称/值”对的集合。不同的语言中，它被理解为对象(object)、记录(record)、结构(struct)、字典(dictionary)、哈希表(hash table)、有键列表(keyed list)，或者关联数组

(associative array)。

(2) 值的有序列表。在大部分语言中，它被理解为数组(array)。

这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON 以一种特定的字符串形式来表示 JavaScript 对象。如果将具有这样一种形式的字符串赋给任意一个 JavaScript 变量，那么该变量会变成一个对象引用，而这个对象就是字符串所构建出来的。

例如，假设需要创建一个 User 对象，并具有用户 ID、用户名和用户 Email 这 3 个属性，可以使用以下 JSON 形式来表示 User 对象：

```
{"UserID":110, "Name":"赵艳铎", "Email":gemm@gmail.com"};
```

如果把这一字符串赋予一个 JavaScript 变量，那么就可以直接使用对象的任一属性了。完整代码如下：

```
<script>var User = {"UserID":110, "Name":"赵艳铎", "Email":":gemm@gmail.com"};
alert(User.Name); </script>
```

借助 ASP.NET AJAX Extension，微软选择 JSON 在服务器和 Ajax 客户端实现数据交换，从而创建支持 AJAX 的 Web 服务。在客户端和服务器端都实现了数据的串行化器和并行化器以使数据按 JSON 的格式交换。网页中的客户端脚本与服务器通过 Web 服务通信层进行通信来访问 Web 服务，该通信层使用 AJAX 技术进行 Web 服务调用，数据在客户端与服务器之间通常采用 JSON 格式进行异步交换。

说明：

默认将 JavaScript 对象序列化为 JSON 格式，使用 JavaScript 的 eval 函数可以进行反序列化操作。但 Web 服务和 ASP.NET 网页中的单个方法可以返回其他格式。可以通过 ScriptMethod 属性来指定方法的序列化格式。对于某个 ASMX 服务，可以设置 ScriptMethod 属性形如[ScriptMethod(ResponseFormat = ResponseFormat.Xml)]以使某个 Web 服务方法返回 XML 数据。

9.2　创建和调用 Web 服务

在.NET Framework 中，可以很容易地创建和使用 Web 服务。与 Web 服务相关的命名空间一共有 3 个。

- System.Web.Services：该命名空间中的类用于创建 Web 服务。
- System.Web.Services.Description：使用该命名空间可以通过 WSDL 描述 Web 服务。
- System.Web.Services.Protocols：使用该命名空间可以创建 SOAP 请求和响应。

9.2.1　WebService 类

要创建 Web 服务，可以从 System.Web.Services.WebService 中派生 Web 服务类。WebService 类提供了对 ASP.NET Application 和 Session 对象的访问。WebService 类的常用属性如表 9-1 所示。

表 9-1　WebService 类的常用属性

属　性	说　明
Application	为当前请求返回一个 HttpApplicationState 对象
Context	返回一个封装 HTTP 特定信息的 HttpContext 对象。可以从中读取 HTTP 标题信息
Server	返回一个 HttpServerUtility 对象。这个类有一些方法，可以进行 URL 编码和解码
Session	返回一个 HttpSessionState 对象，以存储客户端的一些状态
User	返回一个实现 IPrincipal 接口的用户对象。使用这个接口可以得到用户名和身份验证类型
SoapVersion	返回 Web 服务使用的 SOAP 版本。SOAP 版本封装在 SoapProtocolVersion 枚举中

1. WebService 属性

与普通的类继承不同的是，WebService 的子类需要用 WebService 属性来标记，该属性用于向 XML Web 服务添加附加信息，如描述其功能的字符串。这是一个 WebServiceAttribute 类的对象，共有 3 个可选属性，如表 9-2 所示。

表 9-2　WebServiceAttribute 类的属性

属　性	说　明
Description	服务的描述信息，可用于 WSDL 文档
Name	获取或设置 Web 服务名称
Namespace	获取或设置 Web 服务的 XML 命名空间。其默认值为 http://tempuri.org，用于测试，但在开发这个服务前，应修改该命名空间

2. WebServiceBinding 属性

.NET 2.0 给 Web 服务添加了一个新属性 WebServiceBinding。这个属性用于把 Web 服务标记为可交互操作的一致性级别。如表 9-3 所示列出了 WebServiceBindingAttribute 类的一些属性。

表 9-3　WebServiceBindingAttribute 类的常用属性

属　性	说　明
ConformanceClaims	Web 服务的一致性级别可设置为 WsiClaims 枚举的一个值。WsiClaims 有两个值，Web 服务遵循 Basic Profile 1.0 时，其值为 BP10；没有定义任何一致性级别时，其值为 None

(续表)

属　　性	说　　明
EmitConformanceClaims	EmitConformanceClaims 是一个布尔属性，定义了用 ConformanceClaims 属性指定的一致性级别是否应传送给生成的 WSDL 文档
Name	使用 Name 属性可以定义绑定的名称。该名称默认与 Web 服务相同，但要加上 Soap 字符串
Location	Location 属性定义了绑定消息的位置，例如 http://www.timeout.com/Webservice.asmx?wsdl
Namespace	Namespace 属性定义了绑定的 XML 命名空间

3. WebMethod 属性

Web 服务中可以使用的所有方法都必须用 WebMethod 属性来标记。当然，也可以有其他没有用 WebMethod 标记的方法，但这些方法只能在 WebMethod 中调用，而不能在客户机上调用。使用属性类 WebMethodAttribute，就可以在远程客户机上调用方法，并可以定义是否缓存响应，缓存时间有多长以及会话状态是否与指定的参数一起存储等。WebMethodAttribute 类的属性如表 9-4 所示。

表 9-4　WebMethodAttribute 类的属性

属　　性	说　　明
BufferResponse	获取或设置响应是否应缓存的标志。默认值为 true。使用被缓存的响应，仅可以将已完成的软件包传递给客户机
CacheDuration	使用这个属性可以设置结果应缓存的时间长短。如果在这个属性设置的时间段中第二次发出了相同的请求，就返回缓存的值。默认值为 0，这表示结果不缓存
Description	该描述用于给预期的用户生成服务帮助页面
EnableSession	布尔值，表示会话状态是否有效。默认值是 false，因此 WebService 类的 Session 属性不能用于存储会话状态
MessageName	默认状态下，把消息名设置为方法名
TransactionOption	这个属性表示方法的事务处理支持。默认值是 Disabled

4. ScriptService 属性

System.Web.Script.Services.ScriptService 属性用于使用 ASP.NET AJAX 从脚本中调用 Web 服务。ScriptService 属性的主要参数如表 9-5 所示。

表 9-5　ScriptService 类的属性

属　　性	说　　明
ResponseFormat	指定是否将响应序列化为 JSON 或者 XML。默认为 JSON，但是，当方法的返回值是 XmlDocument 时，XML 格式会比较方便
UseHttpGet	表明是否可以使用 HTTP GET 调用 Web 服务方法。由于安全性原因，此项的默认设置为 false
XmlSerializeString	表明包括字符串在内的所有返回类型是否都序列化为 XML，默认为 false，当响应格式设置为 JSON 时，将忽略该属性的值

9.2.2　创建 Web 服务

使用 VWD 创建 Web 服务非常简单，只需选择相应的模板，然后按向导提示操作即可。

与其他所有文档类型一样，VWD 也附带有 Web 服务模板。可以使用“添加新项”对话框来添加 Web 服务。之后可以修改服务，并在 Web 浏览器中使用 ASP.NET 运行库自动创建的标准测试页面测试它。当 Web 服务正确运行时，就可以调用该服务。

下面我们创建一个简单的 Web 服务，该 Web 服务提供了两个方法调用： HelloWord 和 AuthCode。其中 HelloWorld 是默认的，而 AuthCode 用于返回一个验证码图片的二进制数据。

例 9-1：创建 Web 服务，返回由字母和数字组成的验证码数据。

(1) 启动 VWD 2010，新建空网站 Chapter9。

(2) 打开“添加新项”对话框，选择“Web 服务”模板，添加名为 WebService.asmx 的 Web 服务。

(3) 此时在网站的 App_Code 文件夹下会自动生成一个名为 WebService.cs 的文件，同时在网站目录中会生成 WebService.asmx 文件，如图 9-3 所示。

(4) 自动生成的 WebService.cs 文件的代码如下所示：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
/// <summary>
///WebService 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。
// [System.Web.Script.Services.ScriptService]
public class WebService : System.Web.Services.WebService {
    public WebService () {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
}
```

从这段代码中，可以找到前面介绍的 4 个属性：WebService、WebServiceBinding、

ScriptService 和 WebMethod。其中，该 Web 服务提供了一个 HelloWorld 方法，调用该方法将返回字符串 Hello Word。

(5) 无须添加和修改任何代码，即可启用和测试 Web 服务 HelloWorld。这里我们适当修改一下 HellWorld 方法，为其添加一个参数，返回 Hello 加参数值，修改后的代码如下：

```
[WebMethod]
public string HelloWorld(string strName) {
    return "Hello "+strName;
}
```

(6) 编译并启动应用程序，在浏览器中打开 WebService.asmx 文件，显示服务支持的操作，如图 9-4 所示。

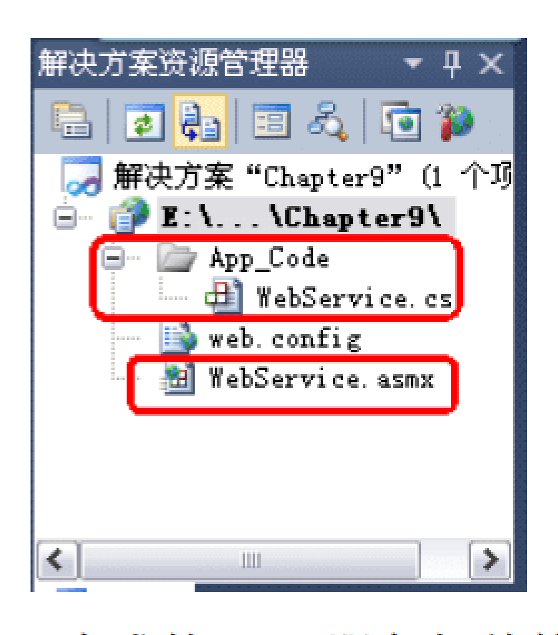


图 9-3　生成的 Web 服务相关的文件

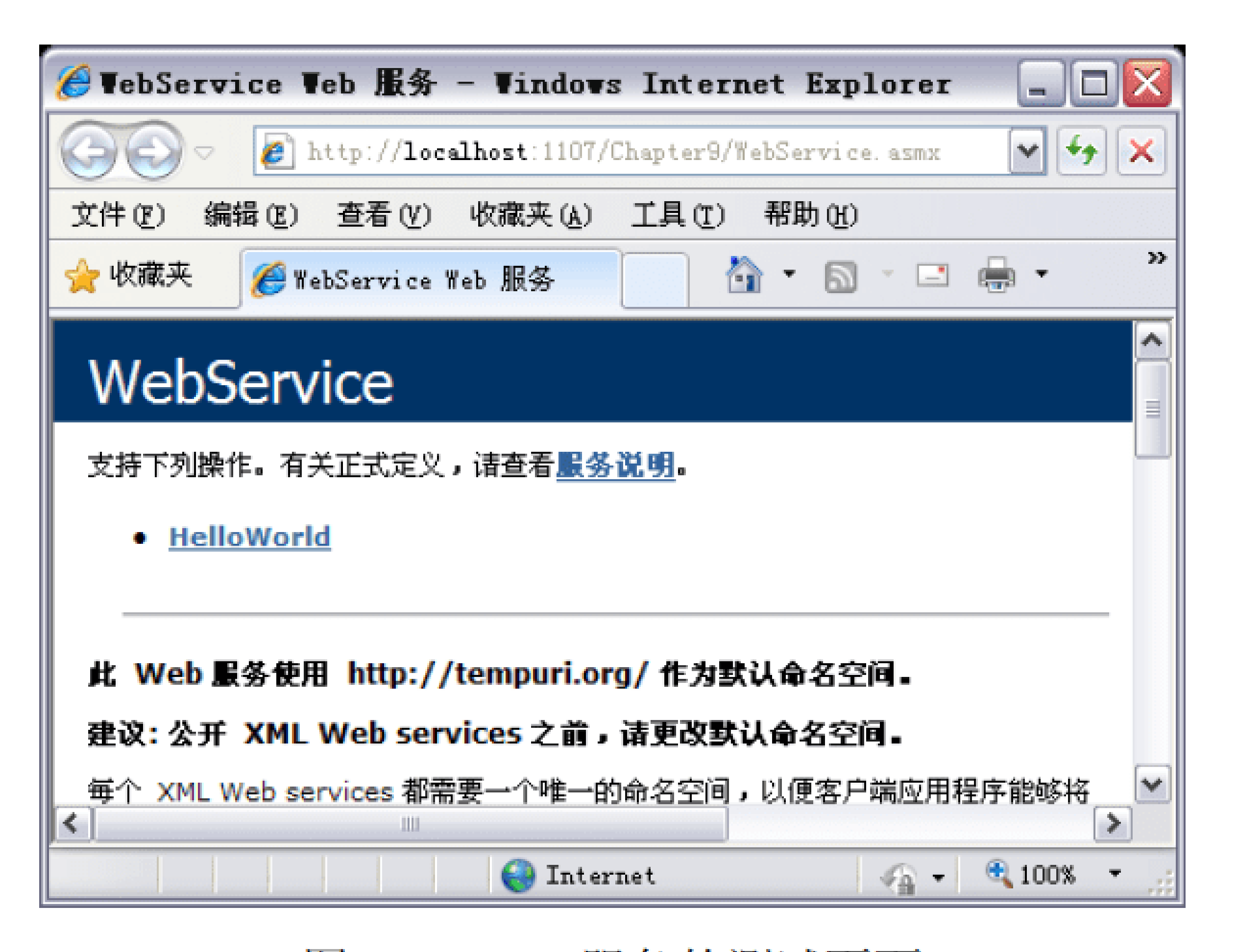


图 9-4　Web 服务的测试页面

(7) 单击 HelloWorld 链接可以调用该方法，如图 9-5 所示。

(8) 因为该方法需要一个字符型的参数 strName，所以在“值”文本框中输入一个参数值后，单击“调用”按钮即可返回调用结果，结果包含在一个 XML 文件中，如图 9-6 所示。

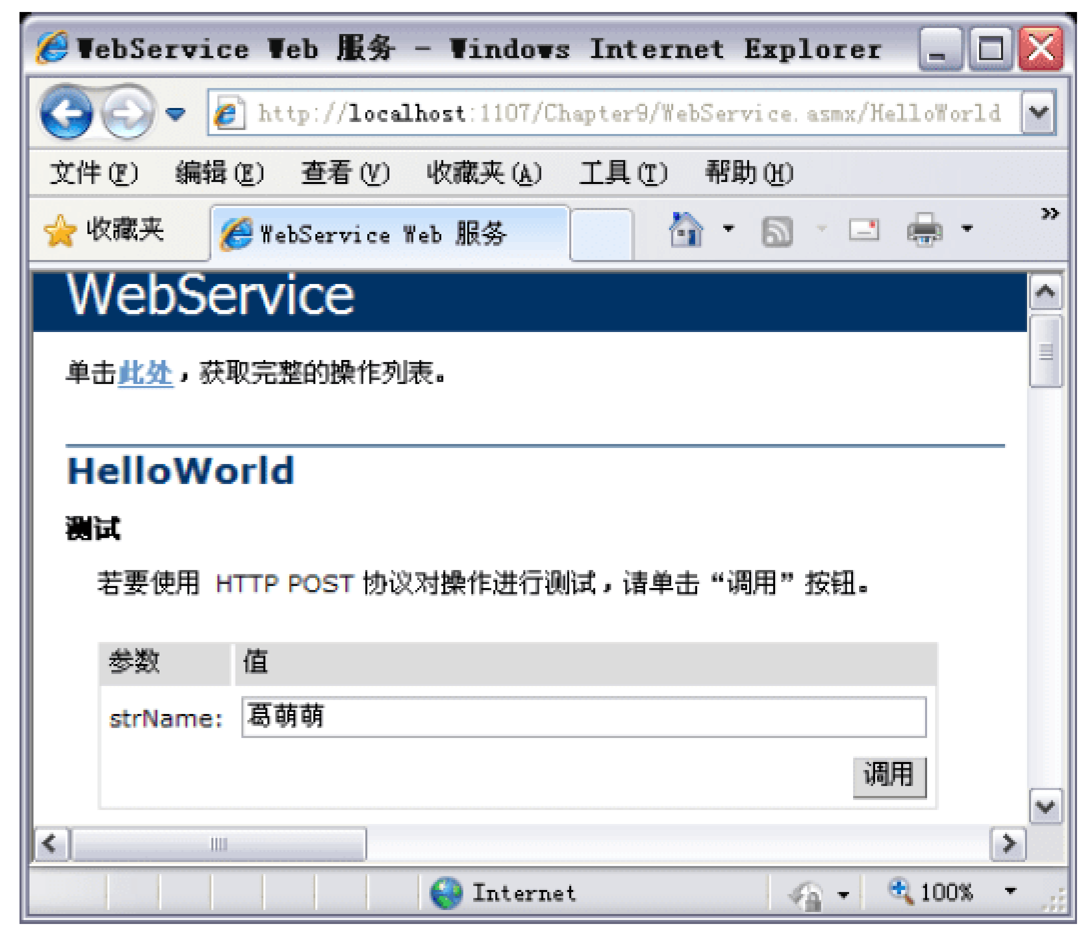


图 9-5　调用 Web 服务

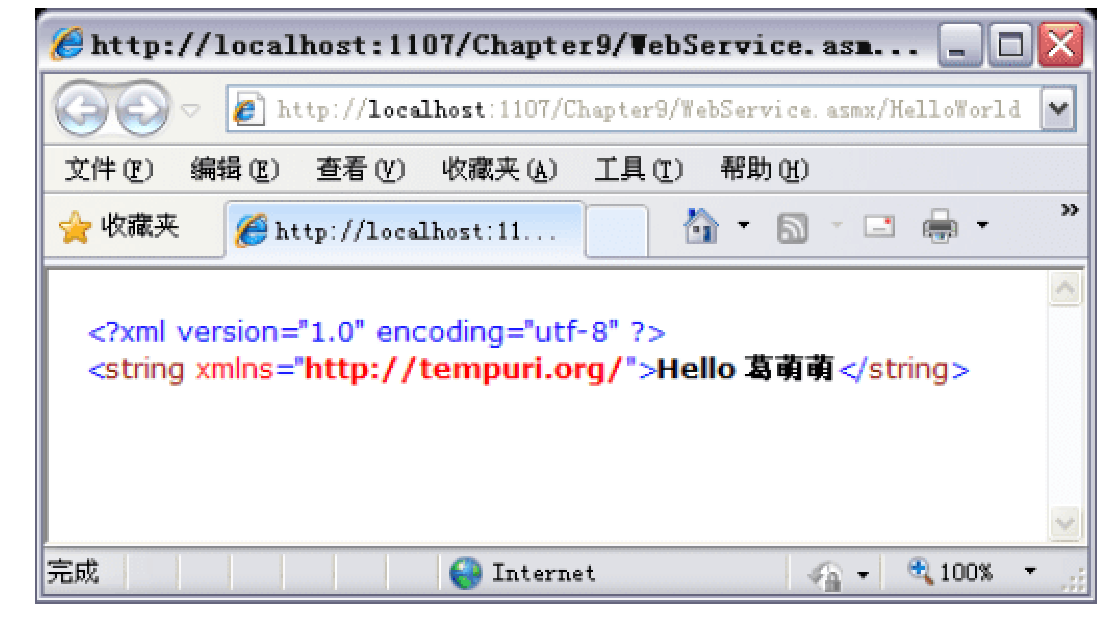


图 9-6　调用 Web 方法返回结果

(9) 返回 WebService.cs 文件，添加绘制验证码的 Web 方法，由于涉及到绘图功能，

所以在要文件头部引入所需的命名空间，代码如下：

```
using System.Drawing;
using System.IO;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
```

(10) Web 方法的定义如下：

```
        [WebMethod]
        public byte[] AuthCode(int length, string checkCode)
        {
            int bmpHeight = 25, bmpWidth = length * 25 + 15; //验证码图片的高度和宽度
            Bitmap bmp = new Bitmap(bmpWidth, bmpHeight);
            int red, blue, green;
            Random rd = new Random(DateTime.Now.Millisecond);
            red = rd.Next(255) % 128 + 128;//产生随机的 RGB 值
            blue = rd.Next(255) % 128 + 128;
            green = rd.Next(255) % 128 + 128;
            Graphics g = Graphics.FromImage(bmp);
            Brush brush = new SolidBrush(Color.FromArgb(red,green,blue));//根据 RGB 颜色值创建
画刷
            g.FillRectangle(brush, 0, 0, bmpWidth, bmpHeight);
            //画图片的前景噪音点
            for (int i = 0; i < 30; i++)
            {
                int x = rd.Next(bmpWidth);
                int y = rd.Next(bmpHeight);
                bmp.SetPixel(x, y, Color.FromArgb(rd.Next()));
            }
            //画图片的边框线
            g.DrawRectangle(new Pen(Color.Silver), 0, 0, bmpWidth - 1, bmpHeight - 1);
            //画图片的背景噪音线
            for (int i = 0; i < 25; i++)
            {
                int x1 = rd.Next(bmpWidth);
                int x2 = rd.Next(bmpWidth);
                int y1 = rd.Next(bmpHeight);
                int y2 = rd.Next(bmpHeight);
                g.DrawLine(new Pen(Color.FromArgb(rd.Next())), x1, y1, x2, y2);
            }
            Rectangle rect = new Rectangle(0, 0, bmpWidth, bmpHeight);
            LinearGradientBrush lgb = new LinearGradientBrush(rect, Color.BlueViolet,
Color.DarkRed, 1.2f, true);
            for (int i = 0; i < length; i++)
```

```
                {//绘制每个字符
                    Font font = new Font("Courier New", 14 + rd.Next() % 4, (FontStyle.Bold |
FontStyle.Italic));
                    g.DrawString(checkCode.Substring(i, 1), font, lgb, 2 + i * 25, 2 + rd.Next(2));
                }
                MemoryStream stream = new MemoryStream();
                bmp.Save(stream, ImageFormat.Gif);
                bmp.Dispose();
                g.Dispose();
                byte[] ret = stream.ToArray();//输出字节流
                stream.Close();
                return ret;
            }
```

上述代码通过 WebMethod 属性指定该方法可以被远程调用。随后的代码是绘制验证码的方法，该方法有两个参数，一个参数是验证码的长度，另一个参数是验证码字符串。

(11) 此时已经完成 Web 服务的创建，可以像刚才那样测试该方法，此时还看不到验证码图片，我们得到的将是通过 BASE64 编码的一个字符串。

9.2.3 节将介绍如何调用 Web 服务，并创建页面调用该方法。

9.2.3　调用 Web 服务

Web 服务的最终目的是提供一种服务接口，由其他程序调用。本节将详细介绍如何调用 Web 服务，包括调用 Web 服务的机制以及如何调用 Web 服务。

1. Web 服务调用机制

调用 Web 服务的第一步就是先找到一个满足需要的 Web 服务。在找到一个服务后，就可以得到这个 Web 服务的描述信息、分组的分类信息和绑定信息。然后根据描述信息，调用相应的方法。

为了找到已经存在的 Web 服务，Microsoft、IBM 和 Ariba 合作建立了一个带有 UDDI 服务的网站 http://www.uddi.org。如果一个公司要发布自己的 Web 服务，就可以在 UDDI 中注册它。有了 UDDI 商务注册表和 UDDI API，就可以编程定位 Web 服务的信息了。

说明：
Web 服务不一定要用 UDDI 注册，也可以从其他资源中获取 Web 服务的信息。

调用 Web 服务的业务流程如图 9-7 所示。服务的描述信息以 Web Services Description Language(Web 服务描述语言，WSDL)格式显示。WSDL 文档描述了 Web 服务支持什么方法、如何调用这些方法、给服务传送的参数类型，以及调用方法返回的参数类型。

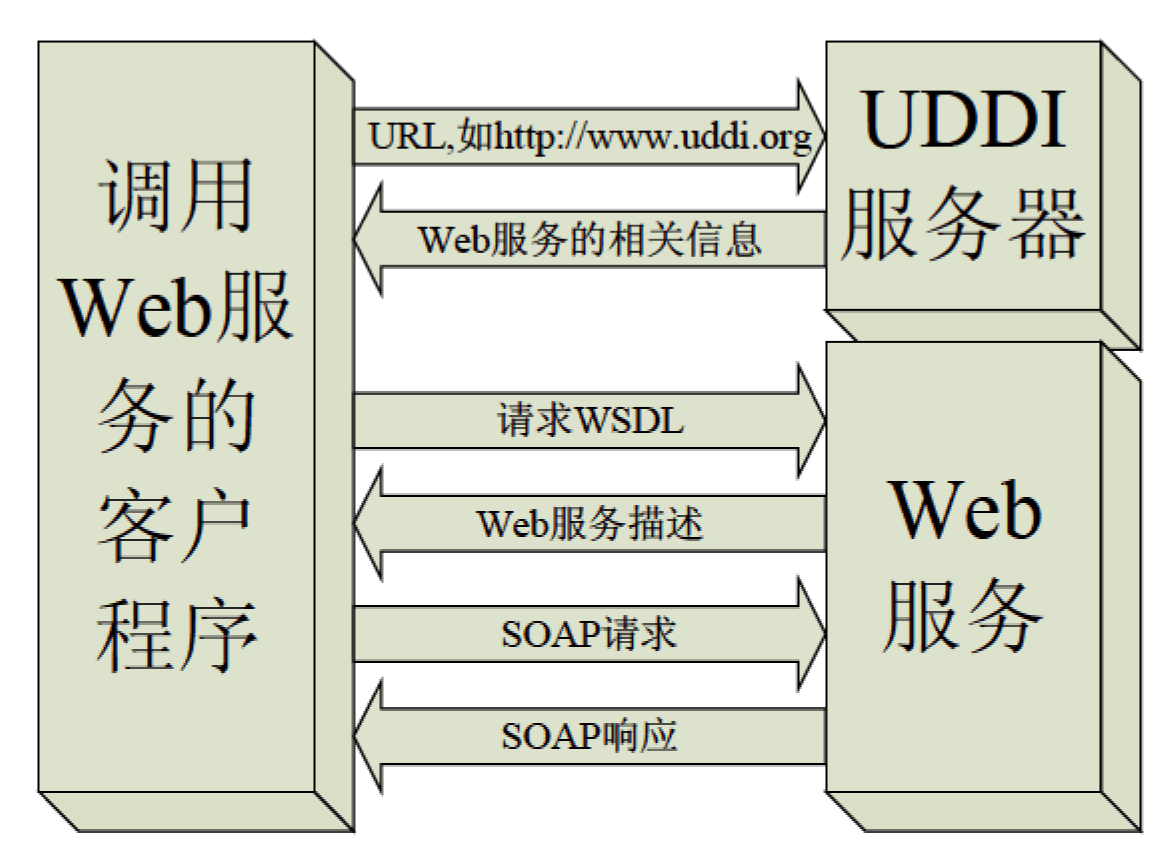


图 9-7　调用 Web 服务的业务流程

技巧：

在.asmx 文件的最后加上字符串?wsdl 就会返回一个 WSDL 文档。WSDL 文档是用 WebMethod 属性动态生成的。

2. 调用 Web 服务

例 9-2：调用例 9-1 中创建的验证码 Web 服务，实现验证码功能。

(1) 启动 VWD 2010，打开网站 Chapter9。

(2) 在“解决方案资源管理器”窗口中，右击解决方案名称，从弹出的快捷菜单中选择“添加服务引用”命令，打开“添加服务引用”对话框，单击“发现”按钮即可搜索到上例中创建的 Web 服务，并显示了当前可用的操作，如图 9-8 所示。

(3) 默认的命名空间为 ServiceReference1，稍候编写代码时会用到这个命名空间，单击“确定”按钮将添加 Web 服务引用，网站目录会自动生成一个名为 App_WebReferences 的文件夹，其中包括一个 ServiceReference1(与前面的命名空间名相同)文件夹，里面有 4 个文件，如图 9-9 所示。

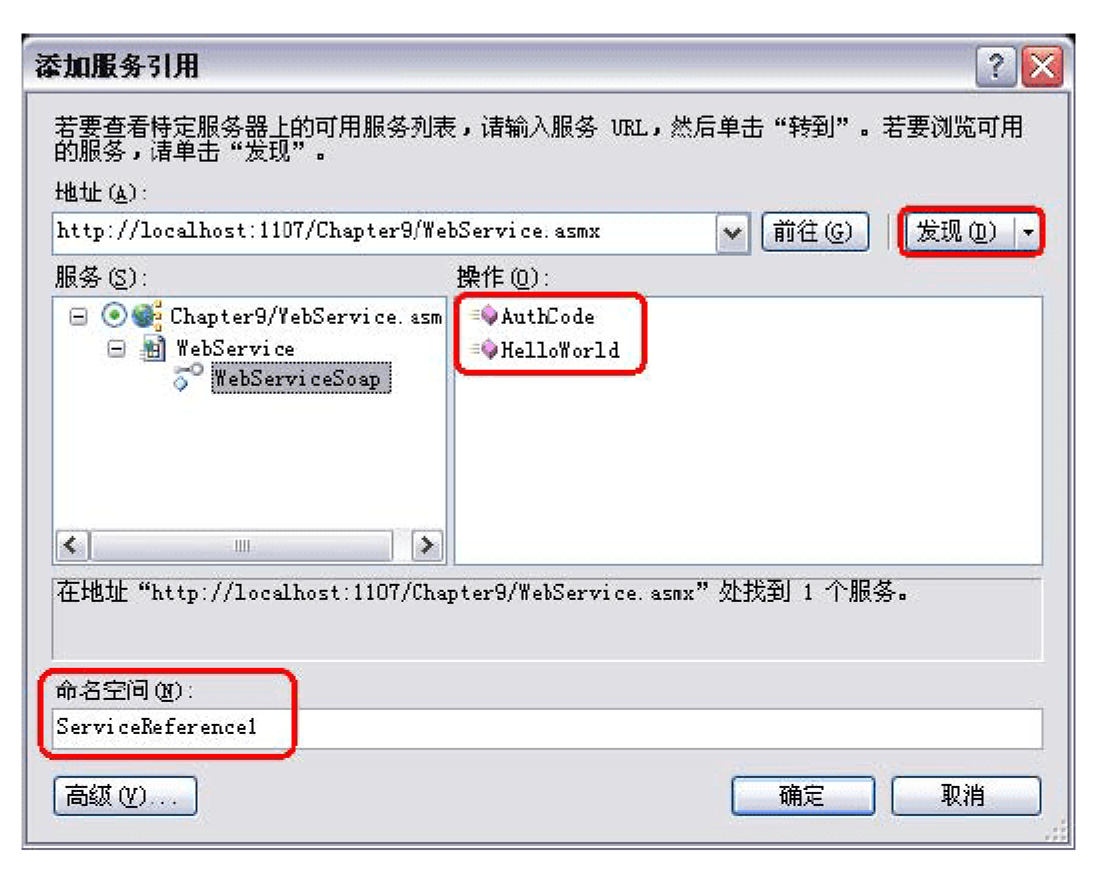


图 9-8　“添加服务引用”对话框

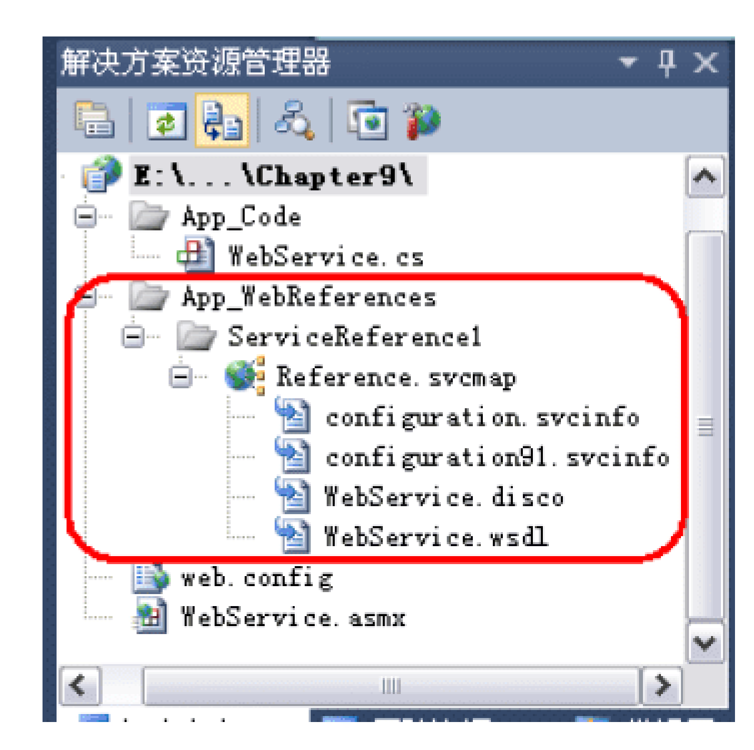


图 9-9　生成的引用文件

(4) 由于 Web 方法 AuthCode 返回的是一个字节数组，为了将这个字节数组表示的图片显示在 Image 控件中，我们需要添加一个 Web 页面，添加一个名为 AuthCode.aspx 的页面。

(5) 在 AuthCode.aspx 的 Load 事件中调用 Web 服务的 AuthCode 方法获取验证码信息，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    ServiceReference1.WebServiceSoapClient c = new
ServiceReference1.WebServiceSoapClient();
    string code = genCode();
    byte[] data = c.AuthCode(code.Length, code);
    Response.ContentType = "image/gif";
    Response.OutputStream.Write(data, 0, data.Length);
}
```

(6) 上述代码首先实例化 Web 服务的对象，然后调用 AuthCode 方法获取验证码图片，验证码是通过 genCode 方法生成的一个算式，相应的代码如下所示：

```
private string genCode()
{
    char[] ops = {'加','减','乘','除' };
    int number1,number2,opNum,result=0;
    string checkCode = String.Empty;
    Random rd = new Random(DateTime.Now.Millisecond);
    number1 = rd.Next(10);//产生第 1 个数字
    number2 = rd.Next(10);//产生第 2 个数字
    opNum = rd.Next(4);//产生操作符对应的数字(0,1,2,3 对应加,减,乘,除)
    switch (opNum)
    {
        case 0:
            checkCode = number1.ToString() + ops[opNum] + number2+"=?";
            result = number1 + number2;
            break;
        case 1:
            if (number1 < number2)
            {
                checkCode = number2.ToString() + ops[opNum] + "?=" + number1;
                result = number2 - number1;
            }
            else
            {
                checkCode = number1.ToString() + ops[opNum] + "?=" + number2;
                result = number1 - number2;
            }
            break;
        case 2:
            checkCode = number1.ToString() + ops[opNum] + number2 + "=?";
```

```
                result = number1 * number2;
                break;
            case 3://除法操作中不能出现 0
                if (number1 == 0)
                    number1 = rd.Next(9)+1;
                if (number2 == 0)
                    number2 = rd.Next(9)+1;
                result = number1 * number2;
                checkCode = result.ToString() + ops[opNum] + "?=" + number2;
                result = number1 ;
                break;
            default:
                break;
        }
        Session["result"] = result;
        return checkCode;
    }
```

在 genCode 方法最后将生成的验证码算式的正确答案保存到 Session 变量中。等用户输入验证码提交后，将从 Session 变量中获得该信息，验证用户的输入是否正确。

(7) 通过“添加新项”对话框添加一个名为 Test.aspx 的测试页面，在 Test.aspx 的源代码视图中添加如下代码：

```
    <form id="form1" runat="server">
    <div>
        验证码：<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Image ID="Image1" runat="server" ImageUrl="~/AuthCode.aspx" />
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Test.aspx">看不清，换
一个</asp:HyperLink>
    </div>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="提交" />
    </form>
```

上述代码中 Image 控件的 ImageUrl 属性指定为 AuthCode.aspx 文件，HyperLink 控件用于刷新页面以便重新获取新的验证码图片。

(8) 在 Button 控件的单击事件处理程序中添加如下代码：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Session["result"].ToString() == TextBox1.Text.Trim())
            Response.Write("<script>alert('验证码输入正确！');</script>");
        else
            Response.Write("<script>alert('验证码输入有误！');</script>");
    }
```

(9) 至此，完成所有代码的编写。编译并运行程序，在浏览器中打开 Test.aspx 页面，效果如图 9-10 所示。

(10) 计算验证码所示算式的结果，输入到文本框中，单击“提交”按钮，将弹出验证成功对话框，如图 9-11 所示。如果看不清验证码图片，则可以单击“看不清，换一个图片”超链接重新获取验证码，如图输入的结果有误，还将弹出相应的提示对话框。

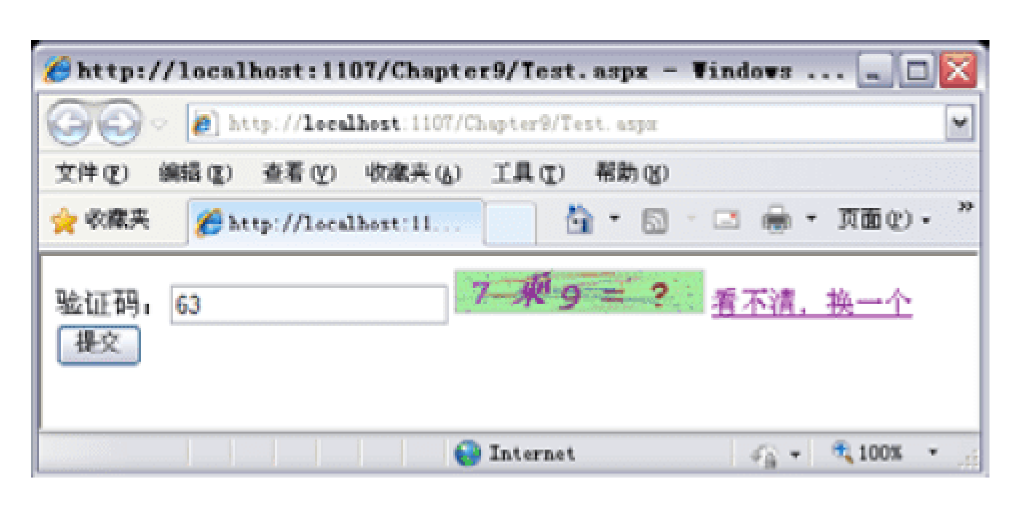




图 9-10　调用 Web 服务获取验证码图片



图 9-11　验证成功对话框

9.3　AJAX 和 Web 服务

典型的 AJAX 体系结构非常容易理解。其工作原理如图 9-12 所示，其中有一个由应用程序特定服务组成的后端，通常只可调用 AJAX 脚本的外层，其下方是业务逻辑所在和发挥作用的系统中间层。服务与前端通过 HTTP 交换数据，使用多种格式传递参数和返回值。前端由运行于客户端上的 JavaScript 代码组成，在接收和处理完数据后，它面临着使用 HTML 和 JavaScript 构建图形用户界面的重大任务。对 JavaScript 的依赖是由于受浏览器结构的限制，只有当浏览器可以支持功能更加强大的编程功能时，这种情况才会改变。

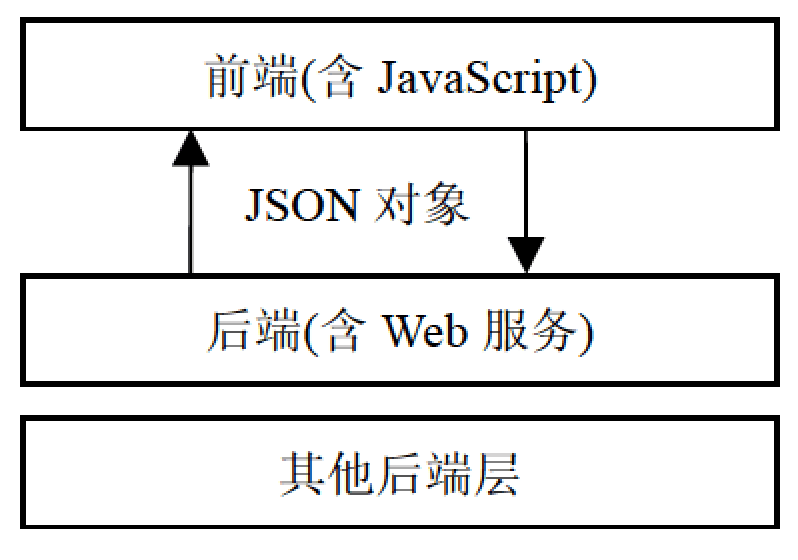


图 9-12　典型的 AJAX 体系结构

本节将介绍如何创建支持 AJAX 的 Web 服务，以及如何在 AJAX 站点中调用页面方法。页面方法是与 Web 服务类似的一种技术，所不同的是，页面方法直接在现有的 ASPX 页面内定义，而不是在单独的 ASMX 文件中定义。且只能从页面运行的脚本中调用页面方法。

9.3.1　创建支持 AJAX 的 Web 服务

最适合 AJAX 应用程序的服务主要涉及向 Web 客户端公开数据和资源。它可以通过 HTTP 获得，并要求客户端使用 URL(也可以是 HTTP 头)访问数据和命令操作。

已经介绍过只需将[System.Web.Script.Services.ScriptService]前面的注释符号删除，即可将整个服务提供为客户端脚本服务。

1. 创建支持 AJAX 的 Web 服务

例 9-3：创建支持 AJAX 的 Web 服务，添加 Web 方法，根据用户登录名从 WeiBo 数据库中查询相应的用户信息。

(1) 启动 VWD 2010，打开网站 Chapter9。

(2) 通过“添加新项”对话框，添加名为 WebService2.asmx 的 Web 服务。

(3) 打开 WebService2.cs 文件，删除[System.Web.Script.Services.ScriptService]属性前面的注释符号。

(4) 由于要访问数据库，所以需要使用 using 语句引入相应的命名空间：

```
using System.Data.SqlClient;
using System.Data.Sql;
```

(5) 为了方便返回用户信息，定义一个结构体 User，代码如下：

```
public struct User
{
    public int user_id;
    public string user_name;
    public string user_login;
    public string user_sex;
    public string user_email;
    public string user_address;
    public string user_telephone;
    public string user_info;
}
```

(6) 将数据库的连接字符串添加到 web.config 文件中的 ConnectionString 中，配置信息如下：

```
<connectionStrings>
    <add name="WeiBoConnectionString" connectionString="Data
Source=zhao\sqlexpress;Initial Catalog=WeiBo;Integrated Security=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

(7) 添加 using 语句引入 System.Web.Configuration;命名空间。

(8) 添加 Web 方法，通过连接数据库查询指定用户的信息，代码如下：

```
[WebMethod]
public User GetUserByLogin(string user_login)
{
    User user = new User();
```

```
            string strConnect =
WebConfigurationManager.ConnectionStrings["WeiBoConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(strConnect);
            con.Open();
            SqlCommand cmd = new SqlCommand("select
user_id,user_name,user_sex,user_email,user_address,"+
                "user_telephone,user_info from Z_USER where user_login = @user_login", con);
            cmd.Parameters.AddWithValue("@user_login", user_login);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())//获取查询结果
            {
                user.user_id = reader.GetInt32(0);
                user.user_name = reader.GetString(1);
                user.user_sex = reader.GetString(2);
                user.user_email = reader.GetString(3);
                user.user_address = reader.GetString(4);
                user.user_telephone = reader.GetString(5);
                user.user_info = reader.GetString(6);
                user.user_login = user_login;
            }
             else
                user.user_id = -1;//表示查询的登录名不存在
           cmd = null;
            con.Close();
            con = null;
            return user;
        }
```

(9) 至此完成 Web 服务的创建，可以在浏览器中加载并测试该服务了。

2. 在 AJAX 站点中使用 Web 服务

在第 8 章介绍过，ScriptManager 控件几乎是所有与 Ajax 相关的操作中必不可少的。它注册客户端 JavaScript 文件，负责使用 UpdatePanel 更新部分页面，可以将 ScriptManager 添加到单个页面中，也可以添加到母版页中，让它变得在整个站点上都可用。

使用 Web 服务时，需要告知 ScriptManager 要给客户端脚本提供 Web 服务。有两种方法可以实现。

- 在母版页中的 ScriptManager 中。
- 在使用 Web 服务的内容页中使用 ScriptManagerProxy 控件。

要在全部或大多数页面中使用 Web 服务，最好是在母版页的 ScriptManager 中声明 Web 服务。给 ScriptManager 控件提供一个<Services>元素，该元素再包含指向公共服务的一个或多个 ServiceReference 元素。

通过在母版页中引用 Web 服务，该 Web 服务将在基于这个母版页的所有页面都可用。这也意味着每个页面都要下载运行这个服务所需的 JavaScript 文件。如果页面根本没有使

用 Web 服务，也会浪费带宽和资源。因此，对于只在一些页面上使用的服务，最好引用页面本身的服务。如果使用的母版页有 ScriptManager 控件，那么在内容页中就要使用 ScriptManagerProxy 控件。下面的例 9-4 将介绍如何在内容页中使用 ScriptManagerProxy 控件来注册 Web 服务。

例 9-4：在 AJAX 站点中的内容页中使用 ScriptManagerProxy 控件调用 Web 服务。

(1) 启动 VWD 2010，打开网站 Chapter9。

(2) 为了演示内容页中使用 ScriptManagerProxy 控件来注册 Web 服务，我们需要先创建一个母版页。通过“添加新项”对话框添加一个母版页 MasterPage.master。

(3) 在母版页中添加一个 ScriptManager 控件。

注意：

ScriptManager 控件必须添加在 ContentPlaceHolder 控件的外部。

(4) 基于母版页 MasterPage.master 创建名为 AjaxWebServiceTest.aspx 的页面。

(5) 在 AjaxWebServiceTest.aspx 页面中添加一个 ScriptManagerProxy 控件，在“属性”面板中设置控件的 Services 属性，这是一个集合属性，单击属性右侧的省略号将打开“ServiceReference 集合编辑器”对话框。单击“添加”按钮，添加一个成员，设置 Path 值为前面创建的 Web 服务 WebService2.asmx，如图 9-13 所示。切换到“源”视图，可以看到生成的代码，如下所示：

```
<asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
    <Services>
        <asp:ServiceReference Path="~/WebService2.asmx" />
    </Services>
</asp:ScriptManagerProxy>
```

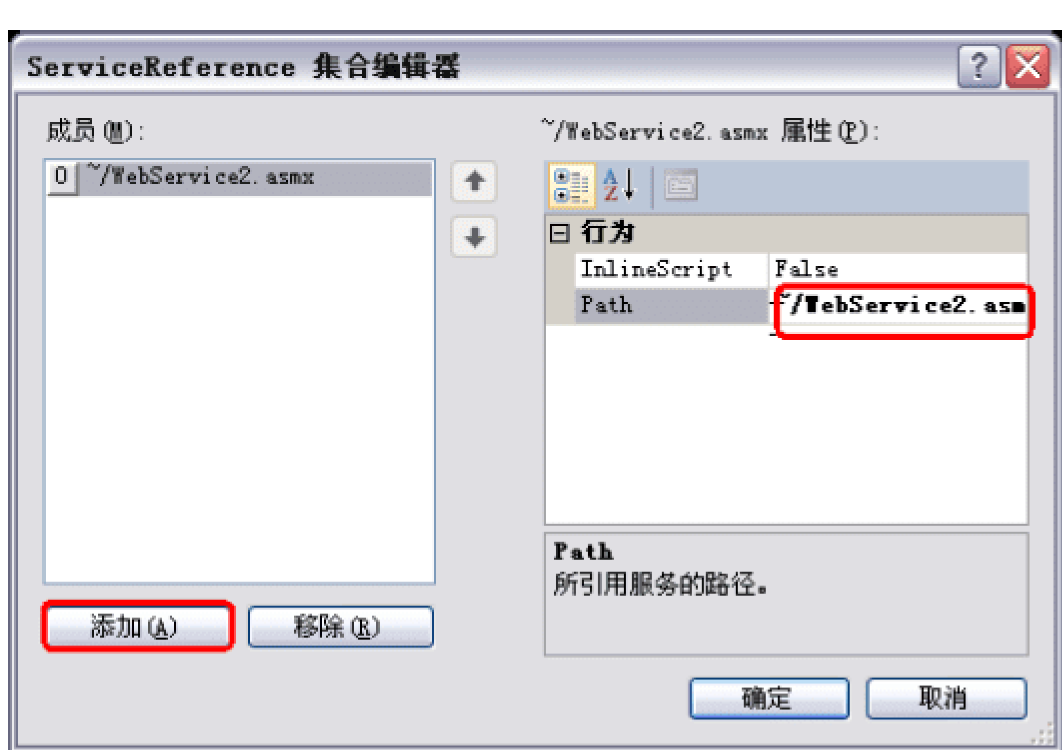


图 9-13　“ServiceReference 集合编辑器”对话框

(6) 在<ScriptManagerProxy>的结束标记下方，添加一个 Input (Text)和一个 Input (Button)，并输入相应的文本提示信息，然后添加一个<div>标记，用于显示返回信息。

```
按登录名查询微博用户信息：<input id="Text1" type="text" />
<input id="Button1" type="button" value="查询" />
<div id="user"></div>
```

(7) 在上述代码的下面添加客户端 JavaScript 代码块：

```
<script type="text/javascript">
    function GetWebMethod() {
        WebService2.GetUserByLogin($get('Text1').value, onSuccCallback, onFailCallback);
    }
    function onSuccCallback(result) {
        var info;
        info = "用户 ID：" + result.user_id;
        info += "<br>姓名：" + result.user_name;
        info += "<br>登录名：" + result.user_login;
        info += "<br>性别：" + result.user_sex;
        info += "<br>Email：" + result.user_email;
        info += "<br>地址：" + result.user_address;
        info += "<br>电话：" + result.user_telephone;
        info += "<br>个人介绍：" + result.user_info;
        if (result.user_id == -1)
            info = "查询的登录名不存在"
        $get('user').innerHTML = info;
    }
    function onFailCallback(error) {
        alert(error.toString);
    }
    $addHandler($get('Button1'), 'click', GetWebMethod);
</script>
```

(8) 编译并运行程序，输入一个登录名，然后单击“查询”按钮，结果如图 9-14 所示。

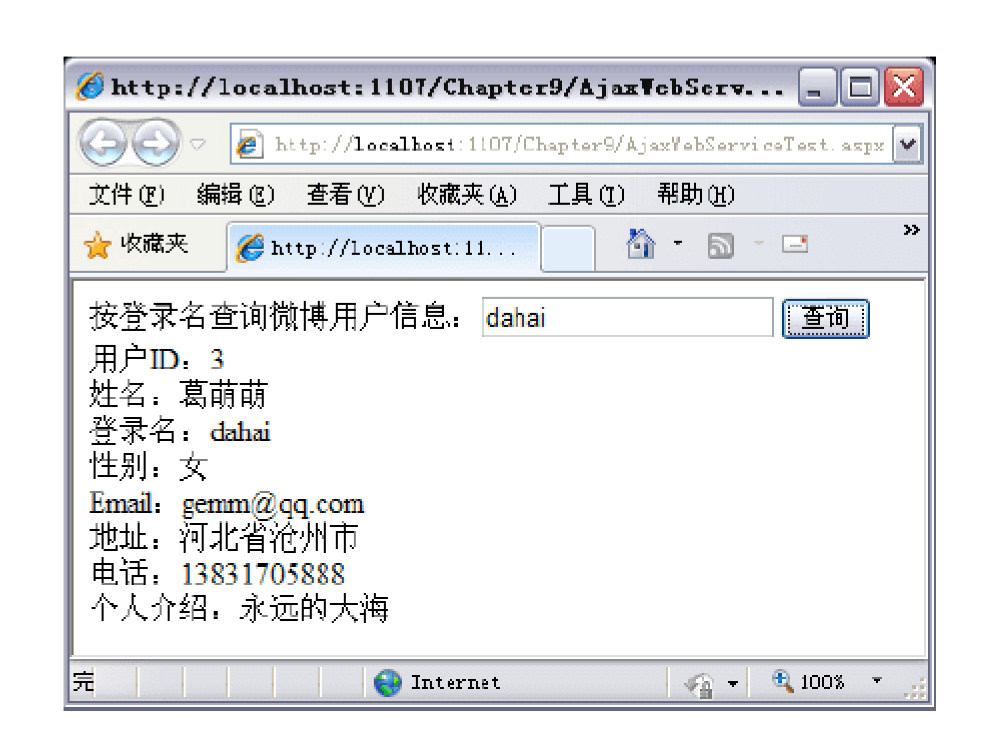


图 9-14　页面运行效果

9.3.2　在 AJAX 站点中调用页面方法

页面方法和 Web 服务有一些共同之处。两者都可以使用很少的代码在客户端调用。可以向它们发送数据，并接收回发的数据。另外，当调用它们时，可以定义成功和失败回调

方法。两者的不同之处在于，页面方法直接在现有的 ASPX 页面内定义，而不是在单独的 ASMX 服务文件中定义。

要启用页面方法，需要将 ScriptManager 控件的 EnablePageMethods 属性设置为 True。ScriptManagerProxy 控件上没有该属性，因此需要直接在 ScriptManager 控件上进行设置。

例 9-5：在 AJAX 站点中调用页面方法。

(1) 启动 VWD 2010，打开建网站 Chapter9。

(2) 通过“添加新项”对话框添加一个名为 WebMethod.aspx 的页面，在页面中添加一个 ScriptManager 控件，设置控件的 EnablePageMethods 属性为 True。

(3) 打开页面的后台代码文件 WebMethod.aspx.cs，添加如下页面方法：

```
[WebMethod]
public static string Welcome(string strName)
{
    return string.Format("你好  {0}", strName);
}
```

提示：

在添加上述代码之前需要首先引入所需的 using System.Web.Services;命名空间。

(4) 切换到 WebMethod.aspx 页面的“源”视图，在</ScriptManager>标记下面添加一个 Input(text)和一个 Input (Button)控件，方法是从工具箱的 HTML 类别中拖动它们。将按钮的 value 设置为“提交”。相应的代码如下：

```
<input id="Text1" type="text" />
<input id="Button1" type="button" value="提交" />
```

(5) 接下来，添加客户端脚本调用页面方法 Welcome，代码如下：

```
<script type="text/javascript">
    $addHandler($get('Button1'), 'click', WebMethodTest);
    function WebMethodTest() {
        var name = $get('Text1').value;
        PageMethods.Welcome(name, SuccCallback);
    }
    function SuccCallback(result) {
        alert(result);
    }
</script>
```

(6) 编译并运行程序，输入一个用户名，单击“提交”按钮，如图 9-15 所示。





图 9-15　调用页面方法允许效果

9.4　本章小结

Web 服务是一个通过 URL 识别的软件应用程序，其界面及绑定能用 XML 文档来定义、描述和发现，使用基于 Internet 协议上的消息传递方式与其他应用程序进行直接交互。Web 服务支持在 Web 站点上放置可编程的元素，使得 Internet 成为一个可以无限扩展、拥有无限潜力的分布式计算平台。利用 Web 服务，公司和个人将能够迅速且廉价地向整个互联网络提供他们的服务。Web 服务具有互操作性、普遍性、松散耦合和高度可集成能力。ASP.NET AJAX 框架会为 Web 服务自动生成 JavaScript 代理，然后通过使用代理来调用 Web 方法。本章详细介绍了 ASP.NET Web 服务体系架构，使用 VWD 创建和调用 Web 服务，以及支持 AJAX 的 Web 服务的创建与调用机制，最后介绍了页面方法的创建与使用。

通过本章的学习，读者应掌握 ASP.NET Web 服务的创建方法和调用机制，能够将自己的程序功能封装成 Web 服务供其他互联网用户使用。

9.5　思考和练习

1. 简述调用 Web 服务的机制和工作原理。
2. 如何创建支持 AJAX 的 Web 服务？
3. 调用天气预报 Web 服务获取国内大城市的天气信息。
4. 创建一个支持 AJAX 的 Web 服务。
5. 页面方法和 Web 服务有何异同点？
6. 登录 http://www.webxml.com.cn/zh_cn/support.aspx，选自己喜欢的 Web 服务，进行编程练习。

第10章　使用jQuery

JavaScript 作为 Web 开发的客户端标准语言，逐渐被人们所重视。但是当掌握 JavaScript 语言的基本用法之后，就会发现 JavaScript 用法比较繁琐，直接使用 DOM 来控制文档，直接调用 XMLHttpRequest 组件来操作数据都显得很麻烦。jQuery 是继 Prototype 之后又一个优秀的 JavaScript 框架，jQuery 能够改变编写 JavaScript 脚本的方式，降低学习和使用 Web 前端开发的复杂度，提高网页开发效率，无论对于 JavaScript 初学者，还是 Web 开发资深专家，jQuery 都应该是必备的工具。本章将主要介绍 jQuery 的基本语法和具体应用。

本章学习目标：

- jQuery 简介
- jQuery 选择器
- jQuery 筛选器
- 使用 jQuery 增强页面
- 使用 jQuery 插件
- jQuery 与 Ajax

10.1　jQuery 简介

jQuery 最早由 John Resig 在 2006 年 1 月开发和发布，现在已经成长为一个备受欢迎的客户端框架。Microsoft 也注意到 jQuery 的强大功能，并决定在自己的产品中附送这个框架。最初，jQuery 随 Microsoft ASP.NET MVC 框架一起提供，现在 Visual Studio 和 Visual Web Developer 2010 中也包含了这个框架。

10.1.1　jQuery 概述

jQuery 具有如下特点。

- 语法简练、语义易懂、学习快速、丰富文档。
- jQuery 是一个轻量级的脚本，其代码非常小巧，最新版的 jQuery 框架文件仅有 30KB 左右。
- jQuery 支持 CSS1~CSS3 定义的属性和选择器，以及基本的 xPath 技术。
- jQuery 是跨浏览器的，它支持几乎所有主流的浏览器，包括 IE 6.0+、FireFox 1.5+、Safari 2.0+和 Opera 9.0+等。
- 可以很容易地为 jQuery 扩展其他功能。

- 能将 JavaScript 脚本与 HTML 源代码完全分离，便于后期编辑和维护。
- 插件丰富，除了 jQuery 自身带有的一些特效外，可以通过插件实现更多功能，如表单验证、Tab 导航、拖放效果、表格排序、DataGrid、树形菜单、图像特效以及 Ajax 上传等。

jQuery 库的主要关注点一直是简化访问 Web 页面元素的方法、帮助处理客户端事件、提供视觉效果(如动画)支持，以及让应用程序中使用 Ajax 变得更加简单。2006 年 1 月，John Resig 公布了 jQuery 的第一版，然后在 2006 年 8 月正式发布了 jQuery 1.0。后来又陆续发布了许多版本，目前最新的稳定版本是 jQuery 1.4.1。

使用 ASP.NET Web 站点模板创建的 Web 站点都包含一个 Scripts 文件夹，其中已经包含了必要的 jQuery 文件。如果基于 ASP.NET 空 Web 站点模板建立网站，也可以手动向 Web 站点添加 jQuery 文件。引入 jQuery 框架文件之后便可在页面脚本中调用 jQuery 对象、方法或属性，并以 jQuery 特色语法规范来编写脚本。

可以从 jQuery 的官方网站 http://jquery.com 上下载 jQuery 的最新版本。该网站不但提供了可以下载的文件，还提供了文档、FAQ、教程和其他有助于更好地利用 jQuery 的信息。

说明：

因为 jQuery 库会增加网页的大小，所以应该明确决定是否在 Web 站点中包含它。

10.1.2　在 Web 站点中引用 jQuery

要在 Web 站点中包含 jQuery，有几种选项可供选择。

- 只在需要 jQuery 的网页或者用户控件中添加对 jQuery 库的引用。这种方式可以有效地减小页面大小。当用户浏览没有使用 jQuery 的页面时，就不需要下载 jQuery 库文件。而当它们下载了库文件以后，浏览器就会缓存库文件的一个副本，从而使得在以后访问页面时，不需要再次下载这些文件。
- 在 Web 站点的母版页中添加对 jQuery 库的引用，从而使所有的页面都可以使用 jQuery 库。这种方式十分方便，因为所有基于该母版页创建的页面都会自动获得对 jQuery 的访问权。但是，这会对 Web 站点第一个页面的性能造成冲击，因为需要从服务器上下载库文件。

由于 jQuery 库很小，所以一般是在母版页中包含它。又由于 jQuery 库由一个使用 JavaScript 代码编写的文件组成，所以可以使用标准的<script>语法对 jQuery 库的引用，例如：

```
<script src=" jquery-1.4.1.min.js" type="text/javascript"></script>
```

必须使用一个独立的结束</script>标记，因为如果使用自结束标记，一些浏览器将无法正常运行代码。

也可以将引用嵌入到 ScriptManager 控件中。ScriptManager 控件有一个<Scripts>子元素，可以用来注册将会添加到浏览器的最后一个页面的 JavaScript 文件。在 ScriptManager 中注册 JavaScript 文件的最简形式如下所示：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
   <Scripts>
      <asp:ScriptReference Path="~/Scripts/jquery-1.4.1.min.js" />
   </Scripts>
</asp:ScriptManager>
```

另外一种方法是使用 Microsoft 的内容传送网络(Content Delivery Network，简称 CDN)或 Google Code 引用 jQuery 库的在线版本。

使用外部库的在线版本的优势在于可以提升服务器的性能并降低带宽。因为站点的访问者很可能已经在访问另外一个站点的时候下载了共享脚本。

10.1.3　jQuery 示例

为了更好地了解 jQuery，下面先来看一个简单的例子。在本例中，将在当前页面中添加 jQuery 库。通过单击按钮来改变表单的背景色。

例 10-1：使用 jQuery 示例。

(1) 启动 VWD2010，新建空网站 Chapter10。

(2) 由于创建的是空网站，所以需要手动添加 jQuery 库。首先在站点的根文件夹中添加一个新的 Scripts 文件夹。

(3) 接下来，将 jQuery 库文件添加到 Scripts 目录中。可以有如下几种方法得到 jQuery 库文件。

- 从 jQuery 网站下载最新的 jQuery 库文件。
- 从本书附带的代码文件夹中找到本章创建的网站，从 Scripts 目录中复制 jQuery 库文件到读者所在的机器中。
- 从 Visual Studio 2010 的安装目录中找到 ProjectTemplatesCache 子目录，这是创建工程时使用的模板目录，依次打开子文件夹 Web/CSharp/2052/WebApplication40.zip/Scripts，该目录下的文件就是 jQuery 库文件。

通过 VWD 模板创建的网站，Scripts 目录中通常有 3 个.js 扩展名的 JavaScript 文件。其中，jquery-1.4.1.js 是完整的 jQuery1.4 库文件，jquery-1.4.1.min.js 是该库文件的简化版本，而 jquery-1.4.1-vsdoc.js 是“智能感知”文档文件，只在 VWD 内使用。

(4) 通过“添加新项”对话框添加一个名为 jQueryTest.aspx 的页面。

(5) 切换到 jQueryTest.aspx 的“源”视图，在<head>标记中添加如下代码引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(6) 在<div>标记中添加 2 个 Input(Button)控件。控件的 value 属性分别为“红色”和“蓝色”。生成的代码如下：

```
    <input id="Button1" type="button" value="红色" />
p   <input id="Button2" type="button" value="蓝色" />
```

(7) 在上述代码的下面添加如下脚本代码：

```
<script type="text/javascript">
    $(document).ready(function () {
        $('#form1').css('background-color', 'green')
        $('#Button1').click(function () {
            $('#form1').css('background-color', 'red')
                .animate({ width: '300px', height: '200px' })
        });
        $('#Button2').click(function () {
            $('#form1').css('background-color', 'blue')
                .animate({ width: '350px', height: '150px' })
        });
    });
</script>
```

提示：

和其他许多编程语言一样，JavaScript(以及 jQuery)对缺少引号、大括号和小括号十分敏感，所以一定要完全按照上面的代码进行输入。在输入代码时，“智能感知”将会弹出，并通过工具提示提供关于各种方法和参数的信息。如果没有弹出“智能感知”，那么可能是没有正确添加<script>元素。

(8) 编译并运行程序，在默认浏览器中打开 jQueryTest.aspx 页面，如图 10-1 所示，表单背景色为绿色，而且仅为一长条。

(9) 单击“红色”按钮，将把背景色设置为红色，大小为(300×200)，如图 10-2 所示。

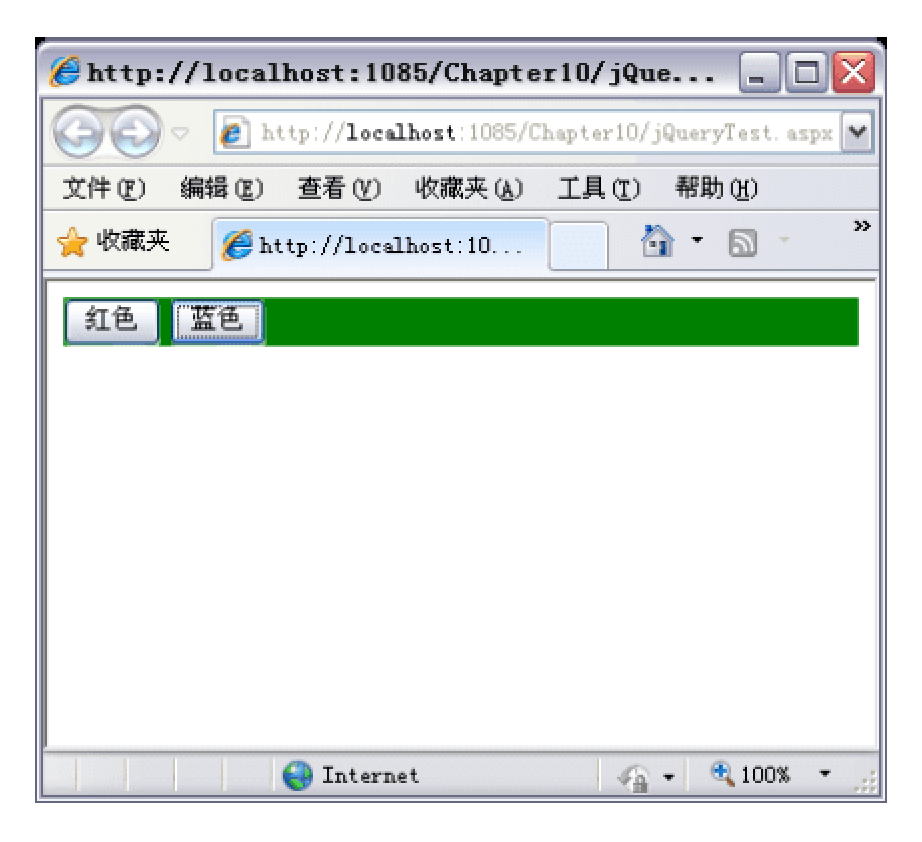


图 10-1　初次加载页面效果

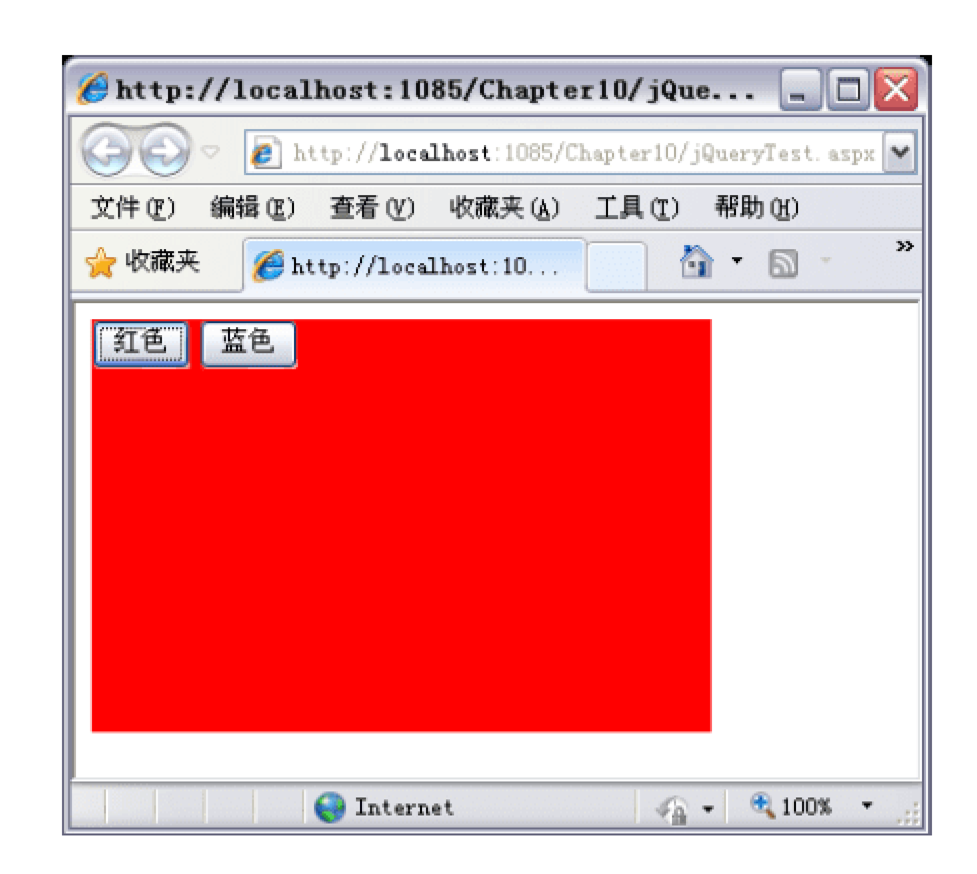


图 10-2　单击“红色”按钮后的效果

上述代码中，添加了一个标准的<script>块，其中可以包含 JavaScript。在这个语句块中，添加了一些在浏览器加载页面完成后触发的 jQuery 代码。页面就绪后，起始大括号({)和结束大括号(})之间的代码将会执行，下面是“文档就绪函数”的示例：

```
<script type="text/javascript">
  $(document).ready(function() {
```

```
        // Remainder of the code skipped
    });
</script>
```

本例读者只需了解 jQuery 的实际应用即可，接下来将详细介绍 jQuery 的语法。

10.2　jQuery 语法

要想理解和使用 jQuery，需要掌握一些基础知识。本节将介绍 jQuery 的核心功能，包括前面看到的$函数，以及$函数的 ready 方法。接下来介绍 jQuery 的选择器和筛选器，这样就可以通过自己指定的条件在页面中查找元素。当获得一个指向页面中一个或多个元素的引用后，就可以对它们应用多种方法，如已经提到过的 css 方法。

10.2.1　ready 函数

大部分 jQuery 代码都是在浏览器完成页面加载后执行。等到页面完成 DOM 加载后再执行代码十分重要。DOM(Document Object Model，即：文档对象模型)是 Web 页面的一种分层表示，包含所有 HTML 元素、脚本文件、CSS 和图像等的一个树形结构。如果借助编程修改 DOM(如使用 jQuery 代码)，那么这种修改将反映在浏览器中显示的页面上。如果过早执行 jQuery 代码(如在页面的最顶端)，那么 DOM 可能还没有加载完成脚本中引用的全部元素时就产生了错误。幸运的是，可以使用 jQuery 中的 ready 函数，将代码的执行推迟到 DOM 就绪。

ready 函数的声明格式如下：

```
$(document).ready(function() {
   // DOM 就绪后执行此处的代码
});
```

当页面准备就绪，可以执行 DOM 操作时，添加到起始和结束大括号之间的全部代码都将执行。jQuery 也提供了 ready 函数的一个快捷方式，下面的代码段与前面的效果相同：

```
$(function() {
   // DOM 就绪后执行此处的代码
});
```

10.2.2　选择器

在 jQuery 中，可以使用美元符号($)作为在页面中查找元素的快捷方式。找到并返回的元素称为匹配集。$方法的基本语法如下所示：

```
$('选择器')
```

在引号(可以使用单引号或者双引号，只要前后两端配对即可)之间，输入一个或多个选择器，接下来就将讨论这方面的内容。

通过 jQuery 选择器可以找到页面的文档对象模型中的一个或多个元素，以便向它们应用各种类型的 jQuery 方法。jQuery 的设计者并没有开发出一种新技术来查找页面元素，而是使用与 CSS 选择器完全相同的选择器。

1. 通用选择器

与对应的 CSS 选择器一样，通用选择器使用通配符*，会匹配页面中的全部元素。$ 方法返回 0 个或多个元素，然后可以使用多种 jQuery 方法操作返回的这些元素。例如，要将页面中所有元素的字体都设置为 Arial，可以使用下面的代码：

```
$('*').css('font-family', 'Arial');
```

2. ID 选择器

和对应的 CSS 选择器一样，这个选择器通过元素的 ID 来查找和获取元素。例如，要为名为 tabel1 的表格设置 CSS 类，可以使用如下代码：

```
$('#table1').addClass('myClass');
```

当这行代码使用 addClass 方法设置 CSS 类时，将会遵循标准的 CSS 规则。即需要通过外部 CSS 文件或者嵌入式样式表定义 myClass 类。

jQuery 的$('#table1')和 ASP.NET AJAX 的$get('table1')都会获得对 ID 为 table1 的一个元素的引用。那么应该选择哪一个方法呢？一般来说，当对结果应用任意 jQuery 方法(例如 css 方法)时，都应该使用 jQuery 的$方法。而在操作单个元素，并且想要修改该元素的某个标准属性时，则可以使用$get 代替。在这种情况下，也可以使用 jQuery 的$，但是因为所有的 jQuery 选择器都返回一个对象集合，所以需要通过索引方式，使用[0]或 get(0)得到第一个元素。下面的 3 个示例具有相同的功能，它们都将 Button1 的 value 值设置为“单击”：

```
$get('Button1').value = '单击';
$('#Button1')[0].value = '单击';
$('#Button1').get(0).value = '单击';
```

3. 元素选择器

元素选择器获得与特定的标记名相匹配的 0 个或多个元素的引用。例如，下面的代码将页面中的所有二级标题的文本颜色设置为蓝色。

```
$('h2').css('color', 'blue');
```

4. 类选择器

类选择器获得与特定的类名相匹配的 0 个或多个元素的引用。例如，如果有下面的 HTML 代码段：

```
<h1 class="Highlight">标题 1</h1>
<h2>标题 2</h2>
<p class="Highlight">如果有一天我不再在乎你了</p>
<p>请记住，曾经也没人听过我的心事</p>
```

上述 4 个元素中有两个元素都有一个名为 Highlight 的 CSS 类。通过 jQuery 的类选择器，可以选择第一个标题和第一个段落，然后将其背景色修改为红色，而保持其他元素不变：

```
$('.Highlight').css('background-color', 'red');
```

5. 分组与合并选择器

和 CSS 一样，可以分组或合并选择器。下面的分组选择器将修改页面中所有 h1 和 h2 元素的文本颜色为橙色：

```
$('h1, h2').css('color', 'orange');
```

通过使用合并选择器，可以找出被其他一些元素包含着的特定元素。例如，下面的 jQuery 只修改 MainContent 元素中包含的三级标题，而保持其他的不变：

```
$('#MainContent h3').css('color', 'red');
```

6. 层级选择

jQuery 支持 4 类层级选择器，分别如下。

- ancestor descendant：在指定祖先元素下匹配所有的后代元素，与 CSS 中的包含选择器对应。
- parent > child：在给定的父元素下匹配所有的子元素，与 CSS 中的子选择器对应。
- prev + next：匹配所有紧接在 prev 元素后的 next 元素，与 CSS 中的相邻选择器对应。
- prev ~ siblings：匹配 prev 元素之后的所有 siblings 元素。

例如，有如下代码：

```
<form>
  <label>姓名:</label>
  <input name="name" />
  <fieldset>
      <label>个人介绍:</label>
      <input name="newsletter" />
 </fieldset>
</form>
<input name="none" />
```

那么，使用层级选择器的 jQuery 代码如下：

```
$("form input")   //返回结果: <input name="name" />, <input name="newsletter" />
$("form > input")   //返回结果: <input name="name" />
```

```
$("label + input")   //返回结果: <input name="name" />, <input name="newsletter" />
$("form ~ input")   //返回结果: <input name="none" />
```

7. 使用选择器

为了理解 jQuery 选择器以及可以对匹配集应用的效果，下面举例来说明。

例 10-2：使用 jQuery 选择器，对匹配集应用 CSS 样式或设置动画效果。

(1) 启动 VWD2010，打开网站 Chapter10。

(2) 通过“添加新项”对话框添加一个名为 Selector.aspx 的页面，切换到页面的“源”视图，在<head>标记中添加如下代码引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在<form>标记中添加如下代码：

```
<form id="form1" runat="server">
<h1>H1  基本选择器</h1>
<div>
<p>这里是段落  1,段落和 slide 层都是红色</p>
<div id="slide">slide 层，下面的段落 2 演示  slideUp  和  slideDown  效果
<p>这里是段落  2,</p></div>
<h2 class="SampleClass">类选择器,5 秒渐渐隐藏</h2>
</div>
</form>
<label id="">Form 后的 Label,演示层级选择</label>
```

(4) 在上述代码的下面添加如下 jQuery 代码：

```
<script type="text/javascript">
    $(function () {
        $('*').css('color', 'Green');
        $('#slide').css('border-bottom', '4px solid black');
        $('h1').bind('click', function () { alert('没有谁不希望自己的爱能得到回应') });
        $('.SampleClass').hide(5000);
        $('#slide,p').css('color', 'red');
        $('#slide p').slideUp('slow').slideDown('slow');
        $("form ~ label").css('color', 'blue');
    });
</script>
```

上述 jQuery 代码，首先使用通用选择器设置所有文本的颜色为绿色，设置 slide 层的下方有一个额外的边框线，接着为<h1>元素绑定一个 click 函数，当单击该元素时，将弹出一个对话框，然后通过类选择器设置<h2>元素的隐藏效果。随后又通过分组选择器将 slide 和 p 的文本颜色设置为红色，使用合并选择器设置 slide 中的 p 元素淡入淡出动画效果，最后通过层级选择设置<form>元素后面的<label>元素的字体颜色为蓝色。

(5) 编译并运行程序，在浏览器中打开 Selector.aspx 页面。运行效果如图 10-3 所示。



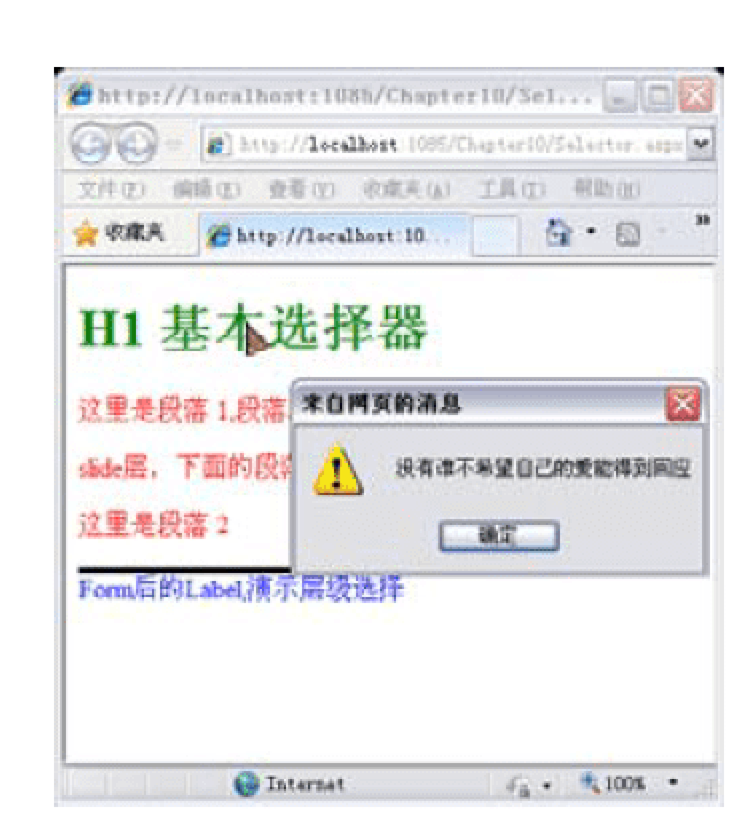


图 10-3　使用选择器页面效果

注意：

添加大量动画会使页面看上去很有趣。但是一般不推荐将所有这些功能同时添加到页面中，否则会吓跑大部分用户。但是，本例做为演示效果却很好，因为可以从中看到 jQuery 的一些强大功能。

在本章后面的部分中，将会看到 jQuery 提供的更多样式和动画方法。现在只需要理解选择器语法，能够在页面中引用元素即可。

10.2.3　筛选器

在 jQuery 中，可以使用筛选器进一步过滤选择器得到的结果集，从而可以找到特定的元素，如第一个元素、最后一个元素、所有奇数行元素、所有偶数行元素、所有的标题或者特定位置的项等。

1. 基本筛选器

如表 10-1 所示为 jQuery 的基本筛选器。

表 10-1　基本筛选器

筛　选　器	用　　途
:first :last	用于选择匹配集中的第一个和最后一个项。下面的示例将表的第一行和最后一行的背景色设置为红色： $('#TableId tr:first').css('background-color', 'red'); $('#TableId tr:last').css('background-color', 'red'); 首先，使用#TableId 找到表。然后使用 tr 找到表的全部行。最后，使用:first 和:last 筛选器找到第一行和最后一行
:odd :even	用于选择匹配集中的奇数行或者偶数行。下面的示例将表的奇数行的背景色修改为红色。因为计数是从零开始的，所以实际上将会看到第二行和第四行的背景色发生了改变(因为它们的索引分别为 1 和 3) $('#TableId tr:odd').css('background-color', 'red');

(续表)

筛　选　器	用　　途

:eq(index)
:lt(index)
:gt(index)
	按照索引匹配元素。:eq(equals)根据索引返回一个元素，而:lt(less than)和:gt(greater than)则分别返回小于或者大于给定索引的项。示例如下：
	$('#TableId tr:eq(0)').css('color', 'green');//修改第一行的颜色
	$('#TableId tr:gt(2)').css('color', 'green');//修改大于第二行的文本颜色
	$('#TableId tr:lt(2)').css('color', 'green');//修改行号小于 2 的行的文本颜色

:header	找到页面中的全部标题(从 h1 到 h6)。示例如下所示： $(':header').css('color', 'green');

要了解更多的基本筛选器，可以阅读 jQuery 文档，网址为 http://api.jquery.com/category/selectors/。

2. 高级筛选器

除了刚才看到的基本筛选器以外，jQuery 还支持其他很多筛选器，它们可以用来根据项包含的文本、是否可见，以及它们包含的任意属性获取项。另外，还有一些筛选器可以获得表单元素(例如按钮、复选框和单选按钮等)，以及大量可以用来选择子元素、父元素、兄弟元素和后代元素的选择器。

如表 10-2 所示为常用的高级筛选器。

表 10-2　高级筛选器

筛　选　器	用　　途
:contains(text)	通过包含的文本匹配元素。下面的代码将使表格中包含“小石头”的单元格颜色为绿色： $('td:contains("小石头")').css('color', 'green'); 如果省略 td，那么整个表都会变成绿色。这是因为表本身也会被匹配(它的一个子表包含文本“小石头”)，所以颜色将应用到整个表上，从而使得每个单元格的文本变为绿色
:has(element)	匹配至少包含一个给出元素的元素。示例如下： $(':header:has("span")').css('color', 'green');//将包含 span 的标题元素颜色修改为绿色
[attribute]	基于给定属性匹配元素。示例如下： $('[type]').css('color', 'green'); 需要在文本框中输入一些文本来查看绿色的字体
[attribute=value]	基于一个属性和该属性的值匹配元素。示例如下： $('[type=text]').css('color', 'green');

(续表)

筛　选　器	用　　途
:input :text :password :radio :checkbox :submit :image :reset :button :hidden :file	这些选择器可以用来匹配特定的客户端 HTML 表单元素。例如，可以使用分组选择器把查找按钮和文本框的代码段重写如下： $(':button, :text').css('color', 'green'); 可以使用这些筛选器来实现一些奇妙的效果。例如，要想编写一些功能来选中一个表单中的全部复选框，可以使用下面的代码： $(':checkbox').attr('checked', true); 要想取消选择全部复选框，可以传递 false 作为 attr 方法的第二个参数

下面举例说明筛选器的用法。借助于这个页面，可以试验本章中的许多示例，读者可自行练习。虽然这些功能很强大，但是如果不能熟练操作它们，那么它们也就没有什么实际价值。10.2.4 节将讨论如何修改匹配集中的项的外观和行为。

例 10-3：使用 jQuery 筛选器。

(1) 启动 VWD2010，打开网站 Chapter10。

(2) 通过“添加新项”对话框添加名为 Filter.aspx 的页面，切换到页面的“源”视图，在<head>标记中添加如下代码引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在<form>标记中添加如下代码：

```
<div>
<h1 title="First Header">一级标题</h1>
<table id="Table1">
   <tr><td>姓名</td><td>电话</td></tr>
   <tr><td>赵艳萌</td><td>15910806516</td></tr>
   <tr><td>葛冰</td><td>13831705809</td></tr>
   <tr><td>小石头</td><td>03172059240</td></tr>
   <tr><td>金百合</td><td>13293426789</td></tr>
</table>
<h2>二级 <span style="font-style: italic; font-weight: bold;">标题</span></h2>
<br />
<input id="Checkbox1" type="checkbox" />80 后
<input id="Checkbox2" type="checkbox" />微博控
<input id="Checkbox3" type="checkbox" />奥运
<input id="Checkbox4" type="checkbox" />娱乐新闻
<input id="Button1" type="button" value="全部选中" />
<input id="Button2" type="button" value="全部取消选中" />
</div>
```

(4) 在<form>元素下方，添加“文档就绪函数”，代码如下：

```
<script type="text/javascript">
    $(function () {
        $('#Table1').attr('border', '1');
        $('#Table1').attr('cellpadding', '2');
        $('#Table1 tr:first').css('background-color', 'red');
        $('#Table1 tr:odd').css('background-color', 'green');
        $(':header').css('color', '#800080');
        $(':header:has("span")').css('border-bottom-style', 'dashed');
        $('#Button1').click(function () {
            $(':checkbox').attr('checked', true);
        });
        $('#Button2').click(function () {
            $(':checkbox').attr('checked', false);
        });
    });
</script>
```

上述代码都是已经学过的选择器和筛选器的应用，请读者细细体会。

(5) 编译并运行程序，在浏览器中打开 Filter.aspx 页面，如图 10-4 所示。单击“全部选中”按钮，可以看到复选框全部被选中，如图 10-5 所示。单击“全部取消选中”按钮将取消所有复选框的选中状态。



图 10-4　使用筛选器页面效果

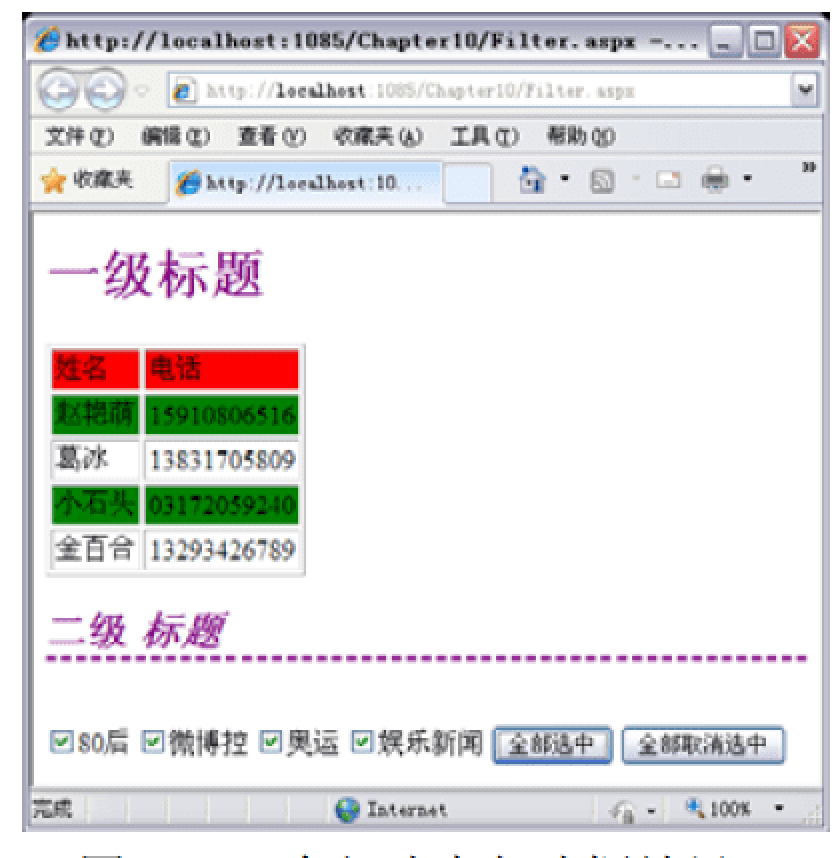


图 10-5　全部选中复选框效果

10.2.4　对匹配集中的项应用 CSS

有了匹配集之后，就需要对它执行一些操作，如前面例子中使用过的 CSS 方法。本节将介绍如何对匹配集中的项应用 CSS 类或者样式。

jQuery 以几种不同的方式支持 CSS。首先，可以使用 CSS 方法来检索特定的 CSS 值(如某个项的颜色)，以及设置一组元素的一个或多个 CSS 属性。其次，使用 addClass、removeClass、toggleClass 和 hasClass 等方法可以修改或检查对元素应用的 CSS 类。再次，

有几种方法可以用来修改元素的尺寸和位置。下面介绍几种常用的方法。

1. css(name,value)

这个方法前面已经用过很多次了，它用来设置某个匹配元素上特定的 CSS 属性。name 参数是引用一个 CSS 属性的名称(如 border、color 等)，value 定义了要应用的样式。下面的代码将修改 h1 元素的背景色为绿色：

```
$('h1').css('background-color', 'green');
```

2. css(name)

这个方法基于传递给它的属性检索特定的 CSS 值。下面的示例将弹出对话框，内容是二级标题中包含的 span 元素的 font-style 属性值。

```
alert($('h2 span').css('font-style'));
```

可以在 jQuery 脚本中使用这个值，例如，可以用来在斜体和普通字体之间切换 font-style，或者将多个元素设置为相同的类型。

3. css(properties)

这是一个功能强大的方法，因为它可以用来同时设置匹配元素的多个属性。下面的语句将把表中所有单元格的颜色修改为红色，将它们的内边距设置为 10px，并设置文本的字体为 Verdana：

```
$('#TableId td').css({'color' : 'red', 'font-family' : 'Verdana',
                'padding' : '10px'});
```

提示：

每个属性和属性值直接由冒号(:)分隔，而每个属性/属性值对之间由逗号分隔。完整的属性集包含在一对花括号({})之间。

4. addClass、removeClass 和 toggleClass

addClass 和 removeClass 方法分别用来在元素中添加和删除类。与普通的 CSS 一样，使用这些方法，比使用 CSS(properties)方法进行内联 CSS 赋值更好。这样就更容易在一个集中的位置定义 CSS 类，从而使得它们更易于维护和重用。

下面的代码将为 h2 元素添加新的 CSS 类：

```
$('h2').addClass('myClass');
```

如果希望再次删除类，则可以调用 removeClass 方法，如下所示：

```
$('h2').removeClass('myClass');
```

如果类还不存在，则 toggleClass 方法将分配一个类；否则，将删除类。

这 3 个方法都允许传递多个类，各个类之间用空格分隔。

结合使用这些 CSS 方法，就在修改页面元素的外观上拥有了极大的控制权。

10.2.5　添加事件处理

事件是许多编程语言中常用的一种技术。在前面几章节中已经看到了.NET 事件的应用，JavaScript 和 DOM 也不例外，它们在许多地方都提供了事件。例如，许多 HTML 元素(如使用 input type="button"定义的按钮)都有一个 click 事件，在单击的时候触发。同样地，它们还有 onmouseover 和 onmouseout 事件，当鼠标经过或者离开它们的时候触发。

即可以在标记中直接定义事件，也可以将事件处理程序定义为一个函数，例如，下面的两种形式都是合法的：

```
<input type="button" onclick="alert('Hello');" value="提交" />
<input type="button" onclick="SayHello();" value="提交" />
```

1. 绑定事件

在学习 AJAX 的时候，已经学过 ASP.NET AJAX 框架的 addHandler 方法，可以在一个独立的代码块中建立处理程序。jQuery 则更进一步，不仅仅允许将事件挂钩到单个元素上，还允许将事件挂钩到整个匹配集上。这种功能极为强大，因为只用几行代码，就可以将处理程序绑定到大量的元素上。例如，为了使表格的外观美观一些，可以设置当鼠标移动到某行时，该行就改变颜色。如果不使用 jQuery，则需要对表的每一行都编写 onmouseover 和 onmouseout 事件。而使用 jQuery，则只需使用如下几行代码：

```
$(function(){
  $('#TableId tr')
        .bind('mouseover', function() { $(this).css('background-color', 'yellow') });
});
```

这些代码将找出#TableId 元素中的全部表行，然后动态分配一个函数，当鼠标悬停在每一行上时，将会调用该函数。

要将 onmouseout 绑定到一个新函数，只需对 bind 的第一次调用返回的值再次调用 bind 即可。jQuery 方法的优点在于，除了应用某些设计或行为，它们会再次返回匹配集。这样就可以对相同的匹配集调用其他方法。这个概念称为链接(chaining)，在这种概念中，使用一个方法的结果作为另外一个方法的输入，从而产生一个效果链。

```
$('#TableId tr')
      .bind('mouseover', function() { $(this).css('background-color', 'yellow') })
      .bind('mouseout', function() { $(this).css('background-color', '') });
```

提示：

上述代码中结束行的分号移动到了最后一行。这样，第二个 bind 就绑定到了前一次对 bind 的调用上。

上述代码完成 3 项工作：首先，使用$('#TableId tr')找出表中的全部行。它在返回的匹配集中调用 bind，以便动态挂钩一些行为，当鼠标移动到某一行上时，就会触发这些行为。然后，对第一次调用 bind 返回的匹配集再次调用 bind，以便当鼠标从该行移走的时候重设背景色。代码中将颜色设置为一个空字符串(' ')，以便删除 CSS 背景属性，这样就可以看到原来的背景。

在这个示例中，还有一个重点地方需要注意，即设置背景色的方式：

```
$(this).css('background-color', 'yellow')
```

其中，this 关键字指的是应用该项的元素，即在本例中就是表行。使用$(this)将得到 jQuery 匹配集(包含单个元素)，可以对其应用常规的 jQuery 方法，如 CSS 方法，也可以不是以 jQuery 而是对 this 元素执行标准的 JavaScript 方法，例如：

```
this.style.backgroundColor = 'yellow'
```

说明：

在 JavaScript 中，短划线(-)不是一个有效的标识符，所有的短划线都将从属性名中删除。而且，原来紧跟在短划线后面的字母将变为大写方式。所以，CSS 中的 background-color 在 JavaScript 中就变成了 backgroundColor，font-family 就变成了 fontFamily 等。

绑定事件之后，也可以使用 unbind([type],[fn])方法删除事件绑定。其中，第一个参数表示要删除绑定的事件名，第二个参数表示删除的附带参数。例如，下面示例将把刚注册的鼠标移走事件删除掉。

```
$('#TableId tr').unbind('mouseout');
```

另外，bind()方法有一个特例就是 one()方法，它能够匹配元素绑定一个仅能够执行一次的事件处理函数。在每个对象上，这个事件处理函数只会被执行一次。其他规则与 bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止(浏览器)默认的行为。例如：

```
$("p").one("click", function(){
  alert( $(this).text() );
});
```

这样，当单击一次<p>元素后，该对象的事件处理函数会自动被注销。

2. 触发事件

trigger 表示开关的意思，jQuery 定义 trigger()方法用来触发默认事件或自定义事件。例如，在下面的示例中自定义了一个事件 myFunc，把该事件绑定到 div 元素上，然后定义事件处理函数，弹出提示对话框，显示 div 元素包含的文本信息。

```
$("div").bind("myFunc", function () {
  alert($(this).text());
});
```

这个自定义事件是无法自动执行的，也不会响应鼠标或键盘行为。但是可以为它定义一个 trigger()方法，代码如下：

```
$("div").trigger("myFunc");
```

将上述 trigger()方法放在 ready 函数中，当页面加载时就会自动执行该自定义事件函数。也可以把这个自定义事件放在另一个事件函数中。这样，只有触发其他事件时，才会响应该自定义事件，并执行自定义事件处理函数。

```
$("div").bind("me", function () {
  alert($(this).text());
});
$("div").bind("mouseover",function(){
  $("div").trigger("me");
});
```

上面脚本将在鼠标移过时，自动触发自定义的事件处理函数。

对于默认事件，如果使用 trigger()方法触发该事件处理函数，同时默认事件自身也可以自动触发事件。例如，在下面这个示例中，当鼠标指针移到 p 元素上时，将触发绑定的事件 click，而当单击 div 元素时，也能够触发该事件绑定的处理函数。

```
<p>做人的遗传比生理的遗传更可怕</p>
<div>没经历过不能体会其中的伤心与无奈</div>
<script language="javascript" type="text/javascript">
$("div").bind("click", function () {
  alert($(this).text());
});
$("p").bind("mouseover",function(){
  $("div").trigger("click");
});
</script>
```

注意：

如果希望仅触发指定事件类型上所有绑定的处理函数，但不执行默认事件，则可以使用 triggerHandler(type,[data])方法。该方法与 trigger()用法相同，但不会触发默认事件。

3. 交互事件

为了简化用户交互操作，jQuery 自定义了两个事件：hover(over,out)和 toggle(fn,fn)。

hover()能够模仿悬停事件，即鼠标移到特定对象上以及移出该对象的方法。它定义当鼠标移到匹配的元素上时，会触发第一个函数。当鼠标移出该元素时，会触发第二个函数。例如，下面的示例定义当鼠标移过段落文本时会设置字体为红色，而移开时又恢复默认颜色。

```
<p>对方并非你的玩具，不是你喜欢时就拿来玩，不喜欢就扔到角落去</p>
<script language="javascript" type="text/javascript">
```

```
            $("p").hover(
                function () {
                    $(this).css("color", "red");
                },
                function () {
                    $(this).css("color", "transparent");
                }
            );
        </script>
```

toggle(fn,fn)能够模仿鼠标单击事件，它表示每次单击时切换要调用的函数。如果单击了一个匹配的元素，则触发指定第一个函数，当再次单击同一元素时，则触发指定第二个函数，随后的每次单击都重复对这两个函数的轮番调用。例如，在上面示例中，将 hover()方法替换为 toggle()方法，这样当单击段落文本时，会自动在默认色和红色之间进行切换。

说明：

对于 toggle 方法可以使用 unbind("click")调用进行删除。

10.2.6　访问 jQuery 对象

通过选择器或筛选器得到的 jQuery 对象是一个集合，要访问该集合，除了使用索引值以外，还可以使用 jQuery 定义的几个方法和属性。另外，jQuery 还扩展并优化了很多筛选函数，这些函数作为 jQuery 对象的方法直接使用，从而在选择器的基础上更加精确地控制对象。

1. each 方法

each 方法迭代(或循环遍历)一个集合。当需要对匹配集中的项应用某种行为，但是无法使用一个 jQuery 函数完成设置时，就可以使用 each 方法。把希望对每一项执行的函数作为参数传递给 each。例如，下面的 each 示例通过循环遍历匹配集中的每一项，然后调用 alert，将每个单元格的内容显示出来。

```
    $('#TableId td').each(function() {
        alert(this.innerHTML);
    });
```

2. size()和 length

size()方法能够返回 jQuery 对象中元素的个数，而 length 属性与 size()方法功能相同。例如，下面的代码使用 size()方法和 length 属性返回值都为 2。

```
    <span>丢了幸福的猪</span>
    <span>我必须接受和容忍你的无知</span>
    <script language="javascript" type="text/javascript">
```

```
        alert($("span").size()); //返回值为 2
        alert($("span").length); //返回值为 2
    </script>
```

3. get 方法

get()方法能够把 jQuery 对象转换为 DOM 中的元素集合。例如，在下面的示例中，使用$()函数获取所有 span 元素，然后用 get()方法把该 jQuery 对象转换为 DOM 集合，再调用 JavaScript 数组方法 reverse()把数组元素的位置颠倒过来。最后为数组中第一个元素设置字体为红色，最终效果是"<span>伤不起<span>"中的文本显示为红色。

```
    <span>爱情买卖</span><span>伤不起</span>
    <script language="javascript" type="text/javascript">
    var spans = $("span").get().reverse(); //把当前 jQuery 对象转换为 DOM 对象并颠倒它们的位置
    spans[0].style.color = "red"; //把当前 jQuery 对象设置为红色
    </script>
```

说明：

也可以使用 get(index)方法获取指定索引值的元素对象。

4. index 方法

index 方法用于获取 jQuery 对象中指定元素的索引值。如果找到了匹配的元素，从 0 开始返回；如果没有找到匹配的元素，则返回−1。

如果不给 index()方法传递参数，那么返回值就是这个 jQuery 对象集合中第一个元素相对于其同辈元素的位置；如果参数是一组 DOM 元素或者 jQuery 对象，那么返回值就是传递的元素相对于原集合的位置；如果参数是一个选择器，那么返回值就是原元素相对于选择器匹配元素中的位置。

例如，在下面的示例中，所有的调用都返回 1。

```
    <ul>
      <li id="girl">漂亮的姑娘就要嫁人了</li>
      <li id="miss">思念是一种很玄的东西</li>
      <li id="pie">千里共婵娟</li>
    </ul>
    <script language="javascript" type="text/javascript">
    $('li').index(document.getElementById('miss'));   //1，返回这个对象在原集合中的索引位置
    $('li').index($('#miss'));   //1，传递一个 jQuery 对象
    $('li').index($('li:gt(0)'));   //1，传递一组 jQuery 对象，返回该对象第一个元素在原集合中的索引
位置
    $('#miss').index('li');   //1，传递一个选择器，返回#miss 在所有 li 中的做引位置
    $('#miss').index();   //1，不传递参数，返回这个元素在同辈中的索引位置。
    </script>
```

5. 筛选函数

jQuery 定义了很多能够从选取对象中过滤部分元素的方法，这些方法是对选择器功能的补充。表 10-3 列出了一些常用的筛选函数。

表 10-3　筛选函数

筛 选 函 数	用　　途
eq(index)	获取指定索引值位置上的元素，索引值从 0 开始
hasClass(class)	检查当前元素是否含有某个特定的类，如果有，则返回 true
filter(expr)	筛选与指定表达式匹配的元素集合。该方法用于缩小匹配范围，用逗号分隔多个表达式
filter(fn)	筛选出与指定函数返回值匹配的元素集合
is(expr)	用一个表达式来检查当前选择的元素集合，如果其中至少有一个元素符合给定的表达式就返回 true
map(callback)	将一组元素转换成其他数组(不论是否是元素数组)
not(expr)	删除与指定表达式匹配的元素
slice(start,[end])	选取一个匹配的子集，与原来的 slice 方法类似
add(expr)	把与表达式匹配的元素添加到 jQuery 对象中。这个函数可以用于连接分别与两个表达式匹配的元素结果集
children([expr])	取得一个包含匹配的元素集合中每一个元素的所有子元素的元素集合
contents()	查找匹配元素内部所有的子节点(包括文本节点)。如果元素是 iframe，则查找文档内容
find(expr)	搜索所有与指定表达式匹配的元素。这个函数是找出正在处理的元素的后代元素
next([expr])	取得一个包含匹配的元素集合中每一个元素紧邻的后面同辈元素的元素集合
nextAll([expr])	查找当前元素之后的所有元素
parent([expr])	取得一个包含着所有匹配元素的唯一父元素的元素集合
parents([expr])	取得一个包含着所有匹配元素的祖先元素的元素集合(不包含根元素)
prev([expr])	取得一个包含匹配的元素集合中每一个元素紧邻的前一个同辈元素的元素集合
prevAll([expr])	查找当前元素之前所有的同辈元素，可以用表达式过滤
siblings([expr])	取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合。可以用可选的表达式进行筛选
andSelf()	加入先前所选的当前元素中，对于筛选或查找后的元素，要加入先前所选元素时将会很有用
end()	回到最近的一个“破坏性”操作之前，即将匹配的元素列表变为前一次的状态

例如，有如下 html 代码：

```
<p><span>早知道伤心总是难免的</span>,你又何苦一往情深</p>
```

那么，$("p").find("span")的结果为：早知道伤心总是难免的。

例 10-4：使用 jQuery 实现图片画廊效果，主要利用 jQurey 选择器和筛选函数选择页面元素，然后再利用 attr()方法读写元素属性实现。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 在网站跟目录添加 images 文件夹，并在该文件夹中添加 5 张图片 image1.jpg~image5.jpg，这 5 张图片就是要用来显示的。

(3) 通过“添加新项”对话框新建样式表 StyleSheet.css，并在该样式文件中添加如下样式定义：

```
body
{ /* 居中显示 */
 text-align: center;
}
#box
{      /* 盒子样式 */
 width: 500px; /* 固定宽度 */
 margin: 12px auto; /* 居中显示 */
}
#largeImg
{/* 大图画框样式 */
 border: solid 1px #FFFF00; /* 定义边框 */
 width: 380px; /* 定义大图画框的宽度 */
 height: 240px; /* 固定高度 */
 padding: 2px; /* 定义补白，增加一点内边距 */
}
.thumbs img
{      /* 缩微图样式 */
 border: solid 1px #C0C0C0; /* 边框样式 */
 width: 50px; /* 宽度 */
 height: 50px; /* 高度 */
 padding: 4px; /* 增加补白 */
}
p
{      /* 段落样式 */
 padding: 0; /* 清除段落补白 */
 margin: 6px; /* 清除段落边界 */
}
```

(4) 通过“添加新项”对话框，添加一个名为 Gallery.aspx 的页面。切换到页面的“源”视图，在<head>标记中添加如下代码引入 jQuery 库和 CSS 样式文件：

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(5) 在<form>标记中添加代码，图片画廊的结构包含在一个 ID 为 box 的 div 内，通过

两个 p 元素分别来组织大图画框和缩微图列表框。代码如下：

```
<div id="box">
    <h1>图片画廊</h1>
    <p><img id="largeImg" src="images/image1.jpg" title="露天电影"alt="图片无法显示" /></p>
    <p class="thumbs">
    <a href="images/image1.jpg" title="露天电影"><img src="images/image1.jpg" /></a>
    <a href="images/image2.jpg" title="一个人发呆"><img src="images/image2.jpg" /></a>
    <a href="images/image3.jpg" title="公交车旁"><img src="images/image3.jpg" /></a>
    <a href="images/image4.jpg" title="高耸云端"><img src="images/image4.jpg" /></a>
    <a href="images/image5.jpg" title="悬空的船"><img src="images/image5.jpg" /></a>
    </p>
</div>
```

(6) 添加 ready 函数定义，其中定义了两个函数：第 1 个函数定义当鼠标指针移过缩微图时，将获取缩微图的地址和提示属性信息，并保存在变量中。然后把这些信息赋值给大图画框内包含的 ID 为 largeImg 的<img>元素，同时设置缩微图的边框颜色，以设计动态效果；第 2 个函数定义在鼠标指针移开缩微图时，恢复缩微图的边框颜色。

```
<script type="text/javascript">
$(function(){// 页面初始化激活函数
    $(".thumbs a").mouseover(function(){ // 鼠标经过缩微图时的处理函数
        var path = $(this).attr("href"); //获取缩微图的地址信息
        var title = $(this).attr("title"); //获取缩微图的提示信息
        $(this).children().css("border-color","#FF9900");   //设置缩微图内超链接样式
        //把缩微图的地址和提示信息赋予给大图
        $("#largeImg").attr({ src:path,title:title});
    });
    $(".thumbs a").mouseout(function(){ // 鼠标移开缩微图时的处理函数
        $(this).children().css("border-color", "#C0C0C0"); // 恢复缩微图的默认边框颜色
    });
});
</script>
```

(7) 编译并运行程序，在浏览器中打开 Gallery.aspx，如图 10-6 所示。滑动鼠标到不同的缩微图，大图将显示不同的图形，如图 10-7 所示。



图 10-6　图片画廊初始页面效果



图 10-7　滑动鼠标显示不同的缩微图

10.2.7　文档处理

可以使用 DOM 为元素节点增加子元素或文本节点，但是 DOM 提供的方法比较烦琐，需要先选中对象，再定义子节点，最后才能使用 appendChild()方法实现插入子元素或文本。jQuery 提供的文档处理方法要比 DOM 简单得多，且功能更为强大和灵活。

1. 插入内容

jQuery 把插入分为内部插入和外部插入两种操作。

所谓内部插入，就是把内容直接插入到指定的元素内部。内部部插入主要包含 6 个方法。

- append(content)：append()方法与 DOM 的 appendChild()方法功能类似，都是在元素内部增加子元素或文本。
- prepend(content)：prepend()方法与 append()方法作用相同，都是把指定内容插入到 jQuery 对象元素中，但是 prepend()方法能够把插入的内容放置在最前面，而不是放置在最末尾。
- appendTo(content)：把所有匹配的元素追加到另一个指定的元素或元素集合中。可以将其理解为 append()方法的反操作，使用这个方法是颠倒了常规的$(A).append(B)的操作，即不是把 B 追加到 A 中，而是把 A 追加到 B 中。
- prependTo(content)：与 appendTo 方法对应，该方法能够把所有匹配的元素前置到另一个指定的元素或元素集合中。
- append(function(index, html))：jQuery 1.4 新增的方法，向每个匹配的元素内部追加内容。这个操作与对指定的元素执行 appendChild 方法，将它们添加到文档中的情况类似。function 函数必须返回一个 HTML 字符串，用于追加到每一个匹配元素的里边。
- prepend(function(index, html))：与 append(function(index, html))对应，也是 jQuery 1.4 新增的方法，向每个匹配的元素内部最前面追加内容。

例如，有如下 HTML 代码：

```
<div></div>
<p>其实你不懂我的心</p>
<p>飞蛾扑火</p>
<b class="foo">我不难过</b>
<b class="foo">等待</b>
```

那么，有如下 jQuery 代码：

```
$("p").append("<b>Hello</b>");
```

结果为：

```
<div></div>
<p>其实你不懂我的心<b>Hello</b></p>
```

```
<p>飞蛾扑火<b>Hello</b></p>
<b class="foo">我不难过</b>
<b class="foo">等待</b>
```

如下 jQuery 代码：

```
$("p").prepend ( $(".foo")[0] );
```

结果为：

```
<div></div>
<p><b class="foo">我不难过</b>其实你不懂我的心</p>
<p><b class="foo">我不难过</b>飞蛾扑火</p>
<b class="foo">我不难过</b>
<b class="foo">等待</b>
```

如下 jQuery 代码：

```
$("p").appendTo("div").addClass("test").end().addClass("test2");
```

结果为：

```
<div>
<p class="test test2">其实你不懂我的心</p>
<p class="test test2">飞蛾扑火</p>
</div>
<p class="test">其实你不懂我的心</p>
<p class="test">飞蛾扑火</p>
<b class="foo test">我不难过</b>
<b class="foo test">等待</b>
```

说明：

appendTo、prependTo、insertBefore、insertAfter 和 replaceAll 这个几个方法成为一个“破坏性”操作，返回值是所有被追加的内容，而不仅仅是先前所选中的元素。所以，要选择先前选中的元素，需要使用 end()方法。

所谓外部插入，就是把内容插入到指定 jQuery 对象相邻元素内。与内部插入操作基本类似，外部插入也包含 6 个方法。

- after(content)：在每个匹配的元素之后插入内容。
- before(content)：在每个匹配的元素之前插入内容。
- insertAfter(content)：把所有匹配的元素插入到另一个指定的元素或元素集合的后面。
- insertBefore(content)：把所有匹配的元素插入到另一个指定的元素或元素集合的前面。
- after(function)：在每个匹配的元素之后插入内容。jQuery 1.4 新增，函数必须返回一个 html 字符串。

- before(function)：在每个匹配的元素之前插入内容。jQuery 1.4 新增，函数必须返回一个 html 字符串。

例如，有如下<div>元素和<p>元素：

```
<p>我的心太乱</p><div id="box">很爱很爱你</div>
```

可以使用如下任意一条 jQuery 代码颠倒两个元素——<div>和<p>元素，排列顺序如下：

```
$("div").after($("p"));
$("p").before($("#box")[0]);
$("p").insertAfter($("div"));
$("div").insertBefore($("p"));
```

上述 4 种方法可以实现相同的功能，但是它们的作用点却各有侧重。

注意：
除了使用 jQuery 对象作为插入内容的参数外，还可以插入 DOM 元素或元素集合，以及 HTML 结构字符串。

2. 嵌套结构

嵌套与插入操作有几分相似，虽然它们都可以实现相同的操作目标，但是两者在概念上还是存在一些区别。嵌套重在结构的构建，而插入则侧重内容的显示。jQuery 定义了如下 9 个嵌套结构的方法。

- wrap(html)：把所有匹配的元素分别用指定结构化标签包裹起来。这种包装对于在文档中插入额外的结构化标记最有用，而且它不会破坏原始文档的语义品质。
- wrap(element)：把所有匹配的元素分别用指定元素包裹起来。
- wrap(function)：把所有匹配的元素用其他元素的结构化标记包装起来。Function 是生成包裹结构的一个函数。
- wrapAll(html)：把所有匹配的元素用一个结构化标签包裹起来。
- wrapAll(element)：把所有匹配的元素用一个元素包裹起来。
- wrapInner(html)：把每一个匹配的元素的子内容(包括文本节点)使用一个 HTML 结构包裹起来。
- wrapInner(element)：把每一个匹配的元素的子内容(包括文本节点)使用元素包裹起来。
- wrapInner(function)：将每一个匹配的元素的子内容(包括文本节点)用 DOM 元素包裹起来，Function 是生成包裹结构的一个函数。
- unwrap()：这个方法将移出元素的父元素。这能快速取消 wrap()方法的效果。匹配的元素以及他们的同辈元素会在 DOM 结构上替换他们的父元素。

例如，对于如下 3 个超链接文本：

```
<a href="~/Index.aspx">首页</a><a href="~/Info.aspx">概述</a><a href="~/About.aspx">关于
</a>
```

如果希望为每个超链接包裹一个<div>标签，则可以使用如下 jQuery 代码：

```
$("a").wrap("<div></div>");
```

最终显示效果的代码结构如下：

```
<div><a href="~/Index.aspx">首页</a></div>
<div><a href="~/Info.aspx">概述</a></div>
<div><a href="~/About.aspx">关于</a></div>
```

如果希望利用页面中现有的元素来包含超链接。例如，如果希望使用超链接底部的列表项元素来包裹超链接文本。

```
<a href="~/Index.aspx">首页</a><a href="~/Info.aspx">概述</a><a href="~/About.aspx">关于
</a>
<ul>
<li> </li>
</ul>
```

可以使用如下 jQuery 代码：

```
$("a").wrap(document.getElementsByTagName("li")[0]);
```

则所得的结果如下所示：

```
<li><a href="~/Index.aspx">首页</a></li>
<li><a href="~/Info.aspx">概述</a></li>
<li><a href="~/About.aspx">关于</a></li>
<ul>
<li> </li>
</ul>
```

如果再为所有列表项外面包一层列表结构 ul 元素，则可以使用如下 jQuery 代码：

```
$("a").wrap(document.getElementsByTagName("li")[0]);
$("li").wrapAll(document.getElementsByTagName("ul")[0]);
```

所得结果如下：

```
<ul>
<li><a href="~/Index.aspx">首页</a></li>
<li><a href="~/Info.aspx">概述</a></li>
<li><a href="~/About.aspx">关于</a></li>
</ul>
<ul>
<li> </li>
</ul>
```

如果希望为每个列表项内嵌入一个 span 元素，则可以使用如下 jQuery 代码：

```
$("a").wrap(document.getElementsByTagName("li")[0]);
$("li").wrapAll(document.getElementsByTagName("ul")[0]);
$("li").wrapInner("<span></span>");
```

所得结果如下：

```
<ul>
<li><span><a href="~/Index.aspx">首页</a></span></li>
<li><span><a href="~/Info.aspx">概述</a></span></li>
<li><span><a href="~/About.aspx">关于</a></span></li>
</ul>
<ul>
<li> </li>
</ul>
```

如果已有如下包裹代码：

```
<div>
    <p>伤不起</p>
    <p>因为爱情</p>
    <p>我不难过</p>
</div>
```

要移除<div>元素，则可以使用如下 jQuery 代码：

```
$("p").unwrap();
```

所得结果如下：

```
    <p>伤不起</p>
    <p>因为爱情</p>
    <p>我不难过</p>
```

3. 替换结构

jQuery 提供了 replaceWith(content)和 replaceAll(selector)方法来实现 HTML 结构替换。

replaceWith()能够将所有匹配的元素替换成指定的 HTML 或 DOM 元素。例如，对于下面 3 个 span 元素，使用 replaceWith()把匹配的所有 span 元素及其包含的文本都替换为“<div>歌曲</div>”。

```
<span>天堂</span><span>红豆</span><span>传奇</span>
<script language="javascript" type="text/javascript">
    $("span").replaceWith("<div>歌曲</div>");
</script>
```

最后，所得到的效果如下：

```
<div>歌曲</div><div>歌曲</div><div>歌曲</div>
```

replaceAll(selector)方法与 replaceWith(content)方法操作正好相反。例如，要实现上面

的替换效果，用 replaceAll 方法就要这样写：

```
$("<div>歌曲</div>").replaceAll("span");
```

4. 删除结构

删除结构有 3 种方法：empty()、remove([expr])和 detach([expr])。

- empty()

使用 empty()可以删除匹配元素包含的所有子节点。例如，在下面示例中将删除 div 元素内所有子节点和文本，返回“<div></div><div></div>”两个空标签。

```
<div>美女</div>
<div><p>为伊消得人憔悴</p></div>
<script language="javascript" type="text/javascript">
    $("div").empty();
</script>
```

- remove ([expr])

使用 remove([expr])方法可以删除匹配的元素，或者符合表达式的匹配元素。例如，在下面示例中将删除 div 元素及其包含的子节点，最后返回的是“<p>下雨收衣服</p>”。

```
<div>你这不孝子</div>
<div><p>打雷了</p></div>
<p>下雨收衣服</p>
<script language="javascript" type="text/javascript">
    $("div").remove();
</script>
```

- detach ([expr])

使用 detach ([expr])方法可以从 DOM 中删除所有匹配的元素。这个方法不会把匹配的元素从 jQuery 对象中删除，因而可以在将来再使用这些匹配的元素。与 remove()不同的是，所有绑定的事件和附加的数据等都会保留下来。例如，在下面示例中将删除具有 hello 类的<p>元素，最后返回的是“昆明湖畔的 <p>小石头吗?</p>”。

```
<p class="hello">还记得</p> 昆明湖畔的 <p>小石头吗?</p>
<script language="javascript" type="text/javascript">
    $("p").detach(".hello");
</script>
```

5. 复制结构

结构复制主要使用 clone()和 clone(true)方法。clone()表示克隆匹配的 DOM 元素并选中克隆的元素。例如，在下面示例中，先使用 clone()方法克隆 div 元素，然后再把它插入到 p 元素内。

```
<div >虚伪的家伙</div>
<p>伪君子</p>
```

```
<script language="javascript" type="text/javascript">
    $("div").clone().appendTo("p");
</script>
```

最后插入结果为：

```
<div>虚伪的家伙</div>
<p>伪君子<div>虚伪的家伙</div></p>
```

clone(true)方法不仅能够克隆元素，而且还可以克隆元素所定义的事件。例如，在上面示例中如果为 div 元素定义一个 onclick 属性事件，则使用 clone(true)方法将会在克隆元素中也包含属性事件。

```
<div onclick="alert('Hello World')"> 虚伪的家伙</div>
<p>伪君子</p>
<script language="javascript" type="text/javascript">
    $("div").clone(true).appendTo("p");
</script>
```

克隆的结果为：

```
<div onclick="alert('Hello World')"> 虚伪的家伙</div>
<p>伪君子<div onclick="alert('Hello World')"> 虚伪的家伙</div></p>
```

10.2.8　使用 jQuery 的效果

在例 10-2 中，使用了 slideUp 和 slideDown 来逐渐地隐藏和显示元素。但是这只是 jQuery 提供的诸多效果和动画方法中的两种方法。本节将介绍其他一些常用的方法，如表 10-4 所示。

表 10-4　常用的动画效果方法

方　法	用　途
show() hide()	通过递减 height、width 和 opacity(使它们变为透明)隐藏或者显示匹配元素。两种方法都允许定义固定的速度(慢、中或快)或者一个定义动画持续时间(单位为毫秒)的数字。例如： $('h1').hide(1000); $('h1').show(1000);
toggle()	toggle 方法在内部使用 show 和 hide 来改变匹配元素的显示方式。即可见元素将被隐藏，不可见元素将会显示。示例如下： $('h1').toggle(2000);
slideDown() slideUp(() slideToggle()	类似于 hide 和 show，这些方法隐藏或显示匹配元素。但是，这是通过将元素的 height 从当前尺寸调整为 0，或者从 0 调整为初始尺寸来实现的。slideToggle 方法会展开隐藏的元素，卷起可见的元素，从而可以使用一个动作重复地显示和隐藏元素。示例如下： $('h1').slideUp(1000).slideDown(1000); $('h1').slideToggle(1000);

(续表)

方　法	用　途
fadeIn() fadeOut() fadeTo()	这些方法通过修改匹配元素的不透明度显示或隐藏它们。fadeOut 将不透明度设置为 0，使元素完全透明，然后将 CSS display 属性设置为 none，从而完全隐藏元素。fadeTo 允许指定一个不透明度(0～1 之间的一个数字)，以便决定元素的透明程度。全部 3 个方法都允许定义一个固定速度(慢、中、快)，或者一个定义了动画持续时间(单位为毫秒)的数字。示例如下： $('h1').fadeOut(1000); $('h1').fadeIn(1000); $('h1').fadeTo(1000, 0.5);
animate()	在内部，animate 用于许多动画方法，如 show 和 hide。但是，也可以在外部使用它，从而可以更灵活地以动画方式显示匹配元素。例如下面这个示例： $('h1').animate({ 　　opacity: 0.4, 　　marginLeft: '50px', 　　fontSize: '50px' 　}, 1500); 这段代码接受一个 h1 元素，将其字体大小设置为 50 像素，将其不透明度设置为 0.4 以使元素半透明，并将其左页边距设置为 50 像素，从而在 1.5 秒的时间内平滑地进行动画显示。animate 方法的第一个参数是一个对象，它保存一个或者多个想要动画显示的属性，每个属性之间以逗号分隔。注意，需要使用 JavaScript 的 marginLeft 和 fontSize，而不是 CSS 的 margin-left 和 font-size 属性。只能动画显示接受数值的属性。也就是说，可以使用 margin、fontSize 和 opacity 等属性，但是不能使用 color 或 fontFamily 这样的属性
stop()	停止所有在指定元素上正在运行的动画，如果队列中有等待执行的动画(并且第一个参数不是 true)，他们将被马上执行
delay()	设置一个延时来推迟执行队列中之后的项目。用于将队列中的函数延时执行。他既可以推迟动画队列的执行，也可以用于自定义队列。例如，下面的代码将在.slideUp()和.fadeIn()之间延时 1 秒： $('h1').slideUp(1000).delay(1000).fadeIn(1000);

下面来看一个 jQuery 实现的动画效果。

例 10-5：使用 jQuery 的效果动态显示图片。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 通过“添加新项”对话框添加一个名为 Animate.aspx 的页面，切换到页面的“源”视图，在<head>标记中添加如下代码引入 jQuery 库和 CSS 样式文件：

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 本例仍然以 images 文件夹中的 5 个图片文件为例进行动态显示。在<form>标记中添加如下代码：

```
<div id="photoShow">
    <div class="photo">
        <img src="images/image1.jpg" />
        <span>露天电影</span>
    </div>
    <div class="photo">
        <img src="images/image2.jpg" />
        <span>一个人发呆</span>
    </div>
    <div class="photo">
        <img src="images/image3.jpg" />
        <span>公交车旁</span>
    </div>
    <div class="photo">
        <img src="images/image4.jpg" />
        <span>高耸云端</span>
    </div>
    <div class="photo">
        <img src="images/image5.jpg" />
        <span>悬空的船</span>
    </div>
</div>
```

(4) 上述代码用到了一些 CSS 样式，所以需要添加这些样式的定义，打开 StyleSheet.css 文件，添加如下代码：

```
#photoShow
{
 border: solid 1px #C5E88E;
 overflow: hidden; /*图片超出 DIV 的部分不显示*/
 width: 490px;
 height: 300px;
 background: #C5E88E;
 position: absolute;
}
.photo
{
 position: absolute;
 top: 0px;
 width: 400px;
 height: 300px;
}
```

```
.photo img
{
 width: 400px;
 height: 300px;
}
.photo span
{
 padding: 5px 0px 0px 5px;
 width: 400px;
 height: 30px;
 position: absolute;
 left: 0px;
 bottom: -32px; /*介绍内容开始的时候不显示*/
 background: black;
 color: #FFFFFF;
}>
```

(5) 添加 jQuery 代码。在 ready 函数中，首先定义了一些变量，然后使用 each()函数为每一个匹配的元素进行事件处理。通过 hover() 函数来处理鼠标的 hover 事件。在这里所有的动画效果都是通过 animate()函数修改 CSS 属性，控制元素的显示位置来实现的。代码如下：

```
    <script type="text/javascript">
        $(document).ready(function () {
            var imgDivs = $("#photoShow>div");
            var imgNum = imgDivs.length; //图片数量
            var divWid = parseInt($("#photoShow").css("width")); //显示宽度
            var imgWid = parseInt($(".photo>img").css("width")); //图片宽度
            var minWid = (divWid - imgWid) / (imgNum - 1); //显示某一张图片时其他图片的宽度
            var spanHeight = parseInt($("#photoShow>.photo:first>span").css("height")); //文字高度
            imgDivs.each(function (i) {
                $(imgDivs[i]).css({ "z-index": i, "left": i * (divWid / imgNum) });
                $(imgDivs[i]).hover(function () {
                    $(this).find("span").stop().animate({ bottom: 0 }, "slow");
                    imgDivs.each(function (j) {
                        if (j <= i) {
                            $(imgDivs[j]).stop().animate({ left: j * minWid }, "slow");
                        } else {
                            $(imgDivs[j]).stop().animate({left:(j-1)*minWid +
imgWid},"slow");
                        }
                    });
                }, function () {
                    imgDivs.each(function (k) {
                        $(this).find("span").stop().animate({ bottom: -spanHeight }, "slow");
```

```
                        $(imgDivs[k]).stop().animate({ left: k * (divWid/imgNum) },"slow");
                    });
                });
            });
        });
    </script>
```

说明：

调用 animate()函数前调用 stop()函数是用来停止当前元素的所有执行中的事件。

(6) 编译并运行程序，在浏览器中打开 Animate.aspx 页面，初始状态如图 10-8 所示。当鼠标进入图片区域后将以大图显示当前鼠标所在的图片，而其他图片将缩小，如图 10-9 所示。

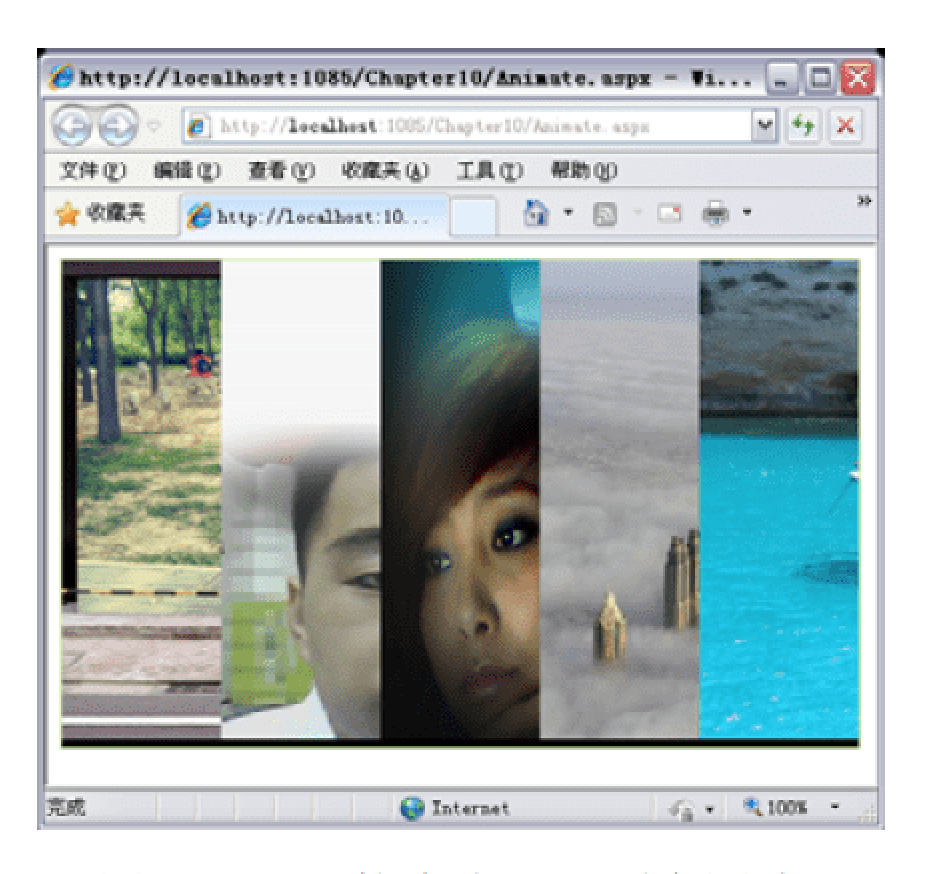


图 10-8　平均大小显示所有图片



图 10-9　鼠标进入图片区域动态显示

虽然一些代码看上去十分复杂，但是使用 jQuery 应用一些特殊效果还是比较简单的。

10.3　jQuery 扩展应用

通过本章的学习，读者已经看到了 jQuery 的强大和优势，但是 jQuery 的功能还远不止如此。jQuery 提供了一个灵活的插件体系结构，使得插件作者编写的功能可以很容易地通过包含一个或多个 JavaScript 文件以及调用一个或者多个方法进行重用。jQuery 社区积极地开发出了数百种实用的插件，可以毫不费力地把它们添加到页面中。

jQuery 插件非常流行，jQuery 站点专门用了一部分空间来提供插件：http://plugins.jquery.com/。除了 jQuery.com 上的插件存储库以外，Internet 上还有其他许多插件。

通常，包含和使用 jQuery 插件的步骤如下。

(1) 在 Internet 上找到并下载要使用的插件。

(2) 在项目中包含插件。即将下载的.js 文件添加到 Scripts 文件夹中。

(3) 在页面中添加对插件文件的引用。如果需要大量使用该插件，则可以把它添加到母版页中。

(4) 编写代码使用插件。具体的代码取决于使用的插件。一般可以在线找到插件的示例或 readme 文件。

10.3.1　使用 jQuery 插件

从 jQuery 网站或 Internet 网站上下载的插件通常都有示例程序，使用这些插件都十分简单，开发人员不必关心它的工作原理，只需参照示例使用插件，执行代码即可实现相应的效果。下面将介绍水印效果的 jQuery 插件的使用，可以在 http://plugins.jquery.com/project/updnWatermark 上找到该插件。

例 10-6：使用水印效果 jQuery 插件。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 将下载的 jQuery 插件解压，将得到的 jquery.updnWatermark.js 文件添加到项目的 Scripts 文件夹中，将 Sample.css 文件添加到项目跟目录中。

(3) 通过"添加新项"对话框添加名为 Watermark.aspx 的页面，切换到页面的"源"视图，在<head>标记中添加如下代码引入 jQuery 库和 CSS 样式文件：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
<script src="Scripts/jquery.updnWatermark.js" type="text/javascript"></script>
<link type="text/css" rel="Stylesheet" href="Sample.css" />
```

(4) 添加一个 TextBox 控件到页面中，要想使插件将水印文本添加到文本框中，需要设置想添加水印的每个控件的 ToolTip 属性。标记中的 ToolTip 属性将映射到最终 HTML 中的 title 属性。设置 TextBox 控件的 ToolTip 属性，并在控件钱添加一些提示性文本，<form>标记中的代码如下：

```
<div>
    <h2>updnWatermark 使用示例</h2>
    姓名：<asp:TextBox ID="TextBox1" runat="server" ToolTip="输入姓名，如 葛萌萌
"></asp:TextBox>
</div>
```

(5) 接下来就是在文档就绪函数中调用插件的主方法，设置控件的水印效果。当输入$.时将弹出"智能提示"窗口，在提示的可用方法列表中可以看到 updnWatermark，如图 10-10 所示。

(6) 可以参照下载的插件的示例代码，添加如下代码：

```
<script type="text/javascript">
    $(function () {
        $.updnWatermark.attachAll();
    });
</script>
```

说明：

不需要选择任何项。只需要调用 jQuery 对象的 updnWatermark 方法(使用$快捷方式)，而不必指定任何选择器。然后 updnWatermark 方法将扫描表单，查找具有 title 属性的表单字段。也可以选择传入一个定义了文本标记的 CSS 类。

(7) 编译并运行程序，运行效果如图 10-11 所示。将光标聚焦到文本框控件上时，水印文本将会消失。不在控件中输入值并失去焦点时，文本将会重新显示。

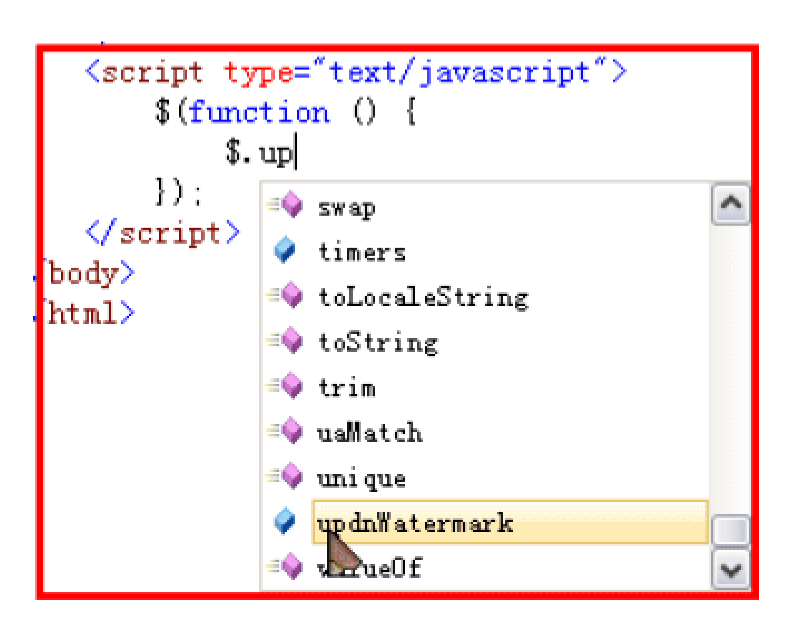


图 10-10　代码的智能提示



图 10-11　水印效果

这个水印插件非常智能，它并不会把文本放到文本框中。相反，它会快速创建一个标记，并将其放到文本框上，从而使文本看上去包含在文本框中。这个解决方案的智能之处在于，因为实际的字段没有改变，所以控件的验证仍然与以前一样。验证控件时会看到一个空文本框，如果没有输入有效数据，但是却想提交表单，验证控件就会触发一个验证弹出窗口。

10.3.2　编写 jQuery 插件

本节将通过一个简单的例子来介绍如何编写 jQuery 插件，要编写的这个 jQuery 插件很简单，实现的功能是给指定表格加上鼠标所在行高亮显示。

1. 编写插件

例 10-7：编写一个给指定表格加上鼠标所在行高亮显示的 jQuery 插件，并提供测试页面。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 在 Scripts 文件夹中通过“添加新项”命令添加一个 JScript 文件 jQuery.tabHighLight.js。

(3) 在新建的文件中添加如下代码：

```
(function ($) {
    $.fn.lightRow = function (row_color) {
        var default_color = "#669900";    //默认颜色
        row_color = (row_color === undefined) ? default_color : row_color;
        $(this).find("tr").each(function () {
            $(this).mouseover(function () {
                $(this).attr("old_color", $(this).css("background-color"));   //创建属性保存旧颜色
```

```
                $(this).css("background-color", row_color);   //使用新颜色
            }).mouseout(function () {
                $(this).css("background-color", $(this).attr("old_color"));    //恢复旧颜色
                $(this).removeAttr("old_color");   //删除保存旧颜色的属性
            });
        });
        return $(this);   //保持操作链
    }
})(jQuery);
```

上述代码的核心部分与例 10-3 中的 jQuery 代码类似，都是添加了鼠标的进入和移出事件，所不同的是头和尾的格式。头部定义了方法的调用方式，方法名是 lightRow 并且可以有一个可选参数指定颜色；尾部定义了方法返回值，返回原调用对象，并在最末尾用(jQuery)标识这是一个 jQuery 插件。

2. 测试插件

例 10-8：添加一个测试页面，引入例 10-7 中创建的 jQuery 插件，并测试插件效果。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 通过“添加新项”命令，添加一个名为的 TestMyjQuery.aspx 页面。

(3) 切换到页面的“源”视图，在<head>标记中添加如下代码引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
<script src="Scripts/jQuery.tabHighLight.js" type="text/javascript"></script>
```

(4) 在页面的<div>元素中添加一个表格，代码如下：

```
<table id="Table1">
  <tr><td>姓名</td><td>电话</td></tr>
  <tr><td>赵艳萌</td><td>15910806516</td></tr>
  <tr><td>葛冰</td><td>13831705809</td></tr>
  <tr><td>小石头</td><td>03172059240</td></tr>
  <tr><td>金百合</td><td>13293426789</td></tr>
</table>
```

(5) 接下来就是在文档就绪函数中调用插件的方法，设置高亮显示鼠标经过行，代码如下：

```
<script type="text/javascript">
    $(function () {
        $('#Table1').attr({border: '1', cellpadding: '2',cellspacing: '2'});
        $('#Table1').lightRow('yellow');
    });
</script>
```

(6) 编译并运行程序，运行效果如图 10-12 所示。

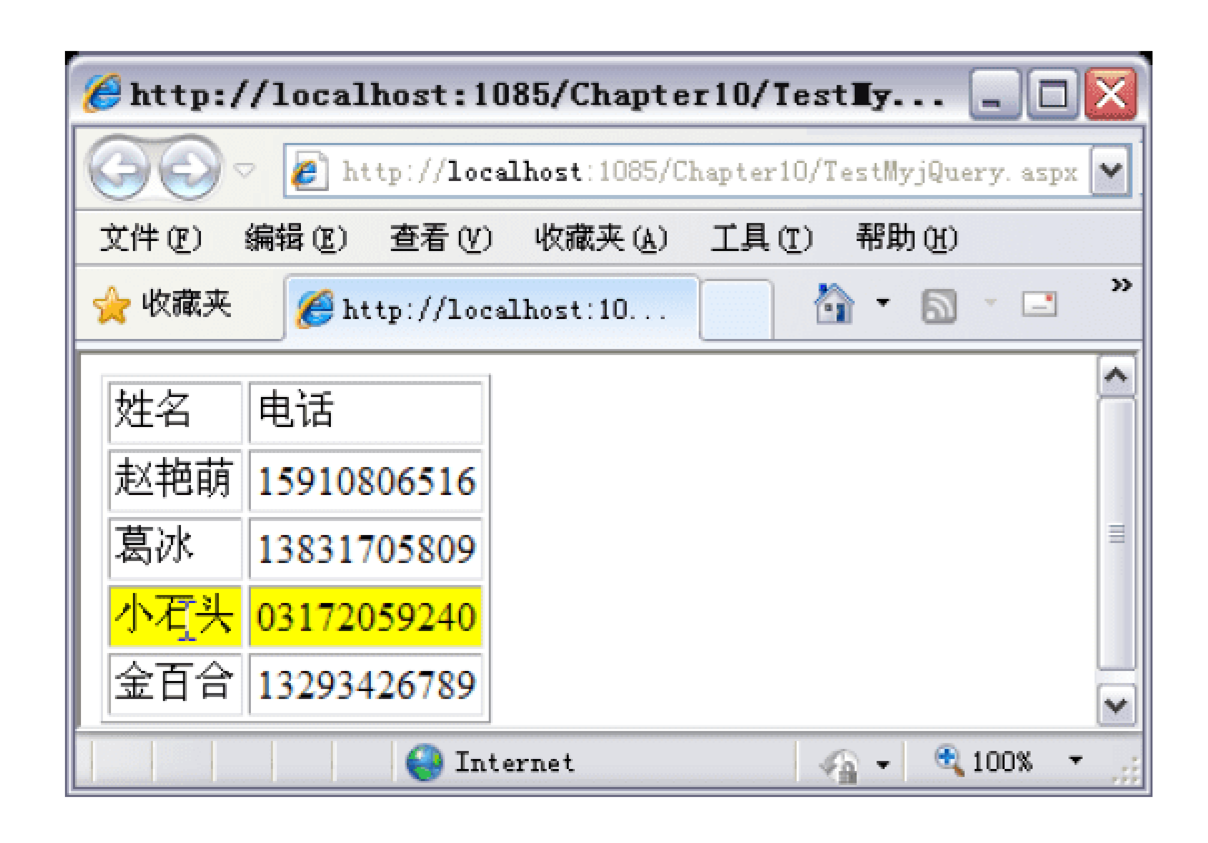


图 10-12　测试自己编写的 jQuery 插件

10.3.3　jQuery 对 Ajax 的支持

本节将介绍 jQuery 对 Ajax 的支持，jQuery 封装了 XMLHttpRequest 组件并初始化，还封装了 Ajax 请求中的各种基本操作，并把这些操作定义为简单的方法。另外，它把 Ajax 请求中各种状态封装为事件，这样只要调用对应的事件就可以快速执行绑定的回调函数。

1. Ajax 请求

jQuery 封装了多种方法实现与远程进行通信，主要有如下几个。

- load(url,[data],[callback])：load()方法能够载入远程 HTML 文件代码并插入到匹配元素中。默认使用 GET 方式，传递附加参数时自动转换为 POST 方式。
- jQuery.get(url,[data],[callback])：jQuery.get()方法能够通过远程 HTTP GET 方式请求载入信息。该方法包含 3 个参数，参数含义与 load()方法相同。
- jQuery.post(url,[data],[callback])：jQuery.post()与 jQuery.get()的操作方法相同，所不同点是它们传递参数的方式不同。jQuery.post()是以 POST 方式来传递参数，所传递的信息可以不受限制，且可以传递二进制信息。

例 10-9：Ajax 请求示例。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 通过“添加新项”对话框添加一个 html 文件 test.htm，在页面的<body>元素中添加如下代码：

```
<body>
    <h2>这是通过 load 方法加载的信息</h2>
    <p>对于梦入膏肓的人来说，揭穿他的梦是残忍的</p>
</body>>
```

(3) 通过“添加新项”对话框添加一个名为 AjaxTest.aspx 的页面，在页面的<head>标记中添加如下代码引入 jQuery 库：

```
    <script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(4) 在 AjaxTest.aspx 页面的<form>标记中添加如下代码：

```
<div id="myDiv"></div>
姓名：<input id="Text1" type="text" /><br />
职业：<input id="Text2" type="text" /><br />
<input id="Button1" type="button" value="Ajax Get" />
<input id="Button2" type="button" value="Ajax Post" />
```

(5) 通过“添加新项”对话框，添加另外两个 Web 窗体，名称分别为 get.aspx 和 post.aspx。

(6) 返回 AjaxTest.aspx 页面，添加如下 jQuery 代码：

```
<script type="text/javascript">
    $('#myDiv').load("test.htm");
    $('#Button1').bind('click', function () {
        $.get("get.aspx", { name: $('#Text1')[0].value, job: $('#Text2')[0].value },
        function (msg) {
            alert(msg);
        }
    );
    });
    $('#Button2').bind('click', function () {
        $.post("post.aspx", { name: $('#Text1')[0].value, job: $('#Text2')[0].value },
        function (msg) {
            alert(msg);
        }
    );
    });
</script>
```

上述代码中，load()方法能够载入远程 HTML 文件并插入到匹配元素中。默认使用 GET 方式，传递附加参数时自动转换为 POST 方式。jQuery 会自动从 text.htm 文档中提取 body 元素包含的代码，并把这些代码插入到匹配的 id 为 myDiv 的<div>元素中。

然后，为两个按钮绑定事件处理程序，分别调用 get 和 post 方法发送请求，并向服务器传递参数，服务器响应之后会把返回值存储在回调函数参数中，弹出消息提示框，这两个按钮提交请求的方式虽然不同，但实现的功能是相同的。

(7) 在 get.aspx 页面中，删除除 Page 指令外的其他代码，添加如下代码：

```
<%
    string str= "通过 GET 方式请求\n 姓名：" + Request.QueryString["name"]+ "\n 职业：" +
Request.QueryString["job"];
    Response.AddHeader("ContentType", "text/html;charset=utf8");
    Response.Write(str);
%>
```

(8) 类似地，在 post.aspx 页面中，也删除除 Page 指令外的代码，添加下面的代码：

```
<%
        string str = "通过 POST 方式请求\n 姓名：" + Request.Form["name"] + "\n 职业：" +
Request.Form["job"];
        Response.AddHeader("ContentType", "text/html;charset=utf8");
        Response.Write(str);
%>
```

(9) 编译并运行程序，在浏览器中打开 AjaxTest.aspx 页面，如图 10-13 所示，在文本框中输入姓名和职业后，单击下面的按钮，将弹出相应的提示对话框，如图 10-14 所示。



图 10-13　页面运行效果



图 10-14　Ajax 调用结果显示

注意：

在使用 load()方法时，所有页面的字符编码应该设置为 utf8，否则 jQuery 在加载文档时会显示乱码。另外，匹配的元素应该只有一个，否则系统会出现异常。

2. jQuery.ajax()请求

jQuery.ajax(options)方法与前面几个方法功能相同，都是向服务器端发送请求，并传递参数，最后调用回调函数获取响应信息。

jQuery.ajax()方法的参数是一个对象包含的参数名/值对组合，其中主要参数名如表 10-5 所示。

表 10-5　jQuery.ajax()方法的主要参数说明

参　数　名	说　　明
async	逻辑值，默认为 true，即请求为异步请求。如果需要发送同步请求，可将其设置为 false。同步请求将锁住浏览器，用户的其他操作必须等待请求完成才可以执行
beforeSend	发送请求前可修改 XMLHttpRequest 对象的函数
cache	是否从浏览器缓存中加载请求信息，默认为 false
complete	请求完成后回调函数，不管请求是成功还是失败均调用
contentType	发送信息到服务器时内容编码类型，默认为适合大多数应用场合
data	发送到服务器的数据将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。如果 processData 选项禁止此自动转换，必须为名/值对格式。如果为数组，jQuery 将自动为不同值对应同一个名称，如：{foo:["bar1","bar2"]}转换为'&foo=bar1&foo=bar2'

(续表)

参　数　名	说　　明
dataType	预期服务器返回的数据类型。取值包括 xml、html、script、json 和 jsonp
error	请求失败时调用函数
global	是否触发全局 Ajax 事件，默认为 true
ifModified	是否仅在服务器数据改变时获取新数据，默认为 false
processData	发送的数据是否可以被转换为对象，默认为 true
success	请求成功后回调函数，参数为服务器返回数据
timeout	设置请求超时时间(单位为毫秒)
type	发送请求的方式，默认为 GET，取值包含 POST、GET、PUT 和 DELETE
url	发送请求的地址

例 10-9 中两个按钮的绑定事件也可以使用如下 jQuery.ajax()请求实现：

```
$('#Button1').bind('click', function () {
    var str = "name=" + $('#Text1')[0].value + "&job=" + $('#Text2')[0].value;
    $.ajax({ url: "get.aspx",
        type: "GET",
        data: str,
        success: function (msg) {
            alert(msg);
        }
    });
});
$('#Button2').bind('click', function () {
    var str = "name=" + $('#Text1')[0].value + "&job=" + $('#Text2')[0].value;
    $.ajax({ url: "post.aspx",
        type: "POST",
        data: str,
        success: function (msg) {
            alert(msg);
        }
    });
});
```

读者可自行上机试验，查看结果是否正确。

3. Ajax 事件

除了 Ajax 请求所使用的方法以外，jQuery 为了方便用户灵活跟踪 Ajax 请求和响应整个完整的过程，还定义了几个事件函数，如表 10-6 所示。

表 10-6　Ajax 事件函数

Ajax 事件	说　　明
ajaxStart(callback)	Ajax 请求开始时执行函数
ajaxSend(callback)	Ajax 请求发送前执行函数
ajaxComplete(callback)	Ajax 请求完成时执行函数
ajaxSuccess(callback)	Ajax 请求成功时执行函数
ajaxError(callback)	Ajax 请求发生错误时执行函数
ajaxStop(callback)	Ajax 请求结束时执行函数

例 10-10：通过 Ajax 事件，跟踪 Ajax 请求的全部过程。

(1) 启动 VWD 2010，打开网站 Chapter10。

(2) 通过“添加新项”对话框添加一个名为 AjaxEvent.aspx 的页面，在页面的<head>标记中添加如下代码引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在 AjaxEvent.aspx 页面的<form>标记中添加与 AjaxTest.aspx 类似的代码：

```
<h2>演示 Ajax 事件</h2><p></p>
姓名：<input id="Text1" type="text" /><br />
职业：<input id="Text2" type="text" /><br />
<input id="Button1" type="button" value="提交" />
```

(4) 在<form>标记下方添加如下 jQuery 代码：

```
<script type="text/javascript">
    $('#Button1').bind('click', function () {
        $.get("get.aspx", { name: $('#Text1')[0].value, job: $('#Text2')[0].value },
        function (msg) {
            alert(msg);
        }
    );
    $('#Button1').ajaxStart(function () {
        $("p").append("Ajax 请求开始<br>");
    });
    $('#Button1').ajaxSend(function () {
        $("p").append("Ajax 请求开始发送<br>");
    });
    $('#Button1').ajaxComplete(function () {
        $("p").append("Ajax 请求完成<br>");
    });
    $('#Button1').ajaxSuccess(function () {
        $("p").append("Ajax 请求成功<br>");
    });
</script>
```

(5) 编译并运行程序，在浏览器中打开 AjaxEvent.aspx 页面，此时还没有 Ajax 事件触发，所以<p>标记中没有显示任何信息，如图 10-15 所示。

(6) 在文本框中输入姓名和城市后，单击“提交”按钮，将弹出提示对话框，此时触发了 ajaxStart 和 ajaxSend 事件，如图 10-16 所示。



图 10-15　页面初始效果

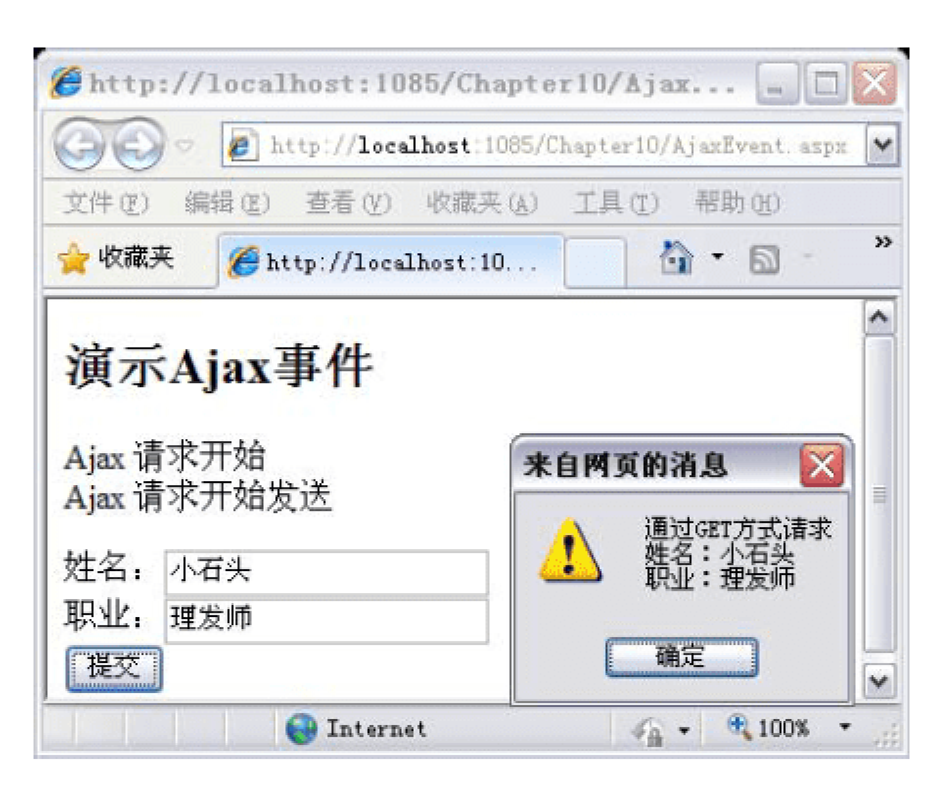


图 10-16　触发了 ajaxStart 和 ajaxSend 事件

(7) 单击“确定”按钮，关闭提示对话框，此时触发了 ajaxSuccess 和 ajaxComplete 事件，如图 10-17 所示。

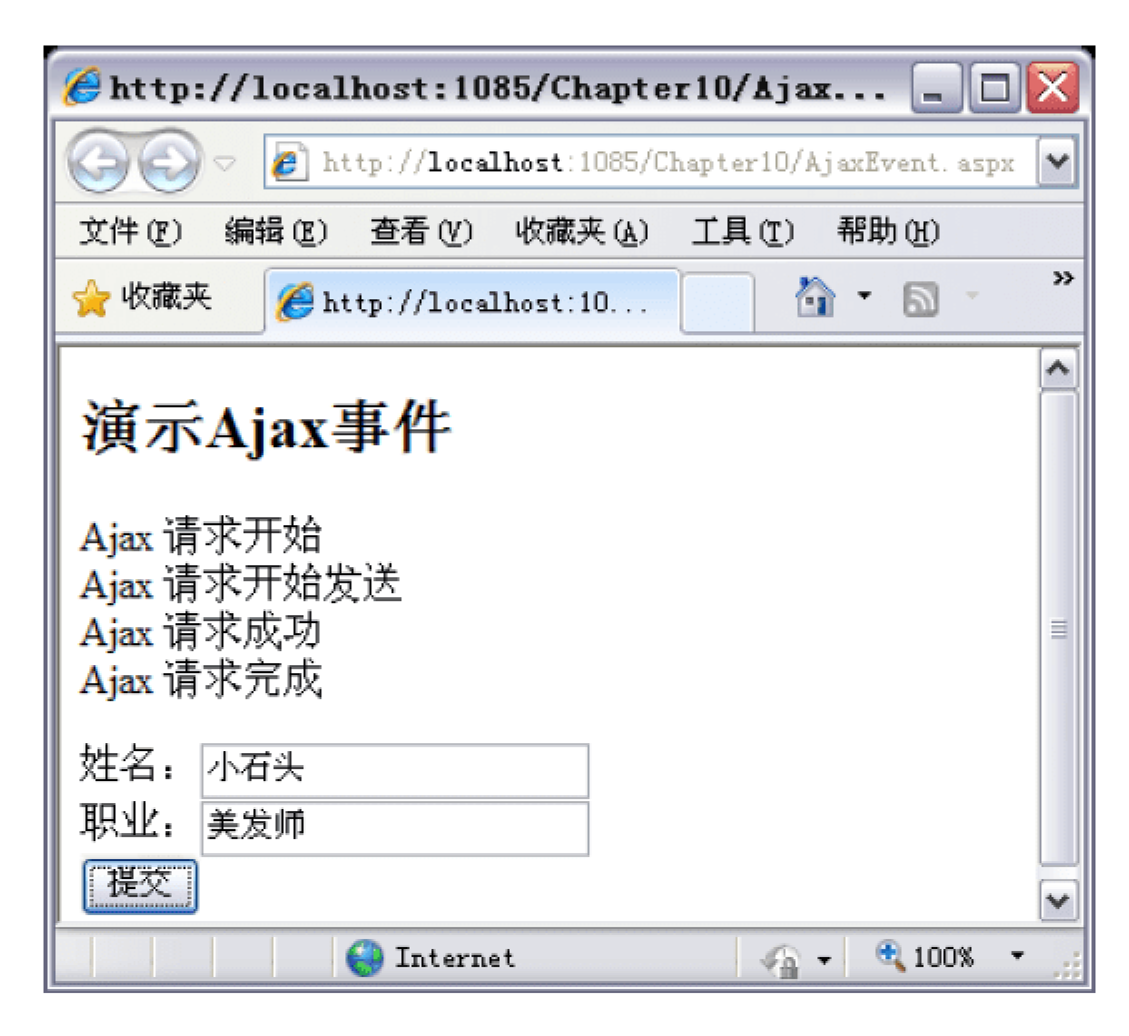


图 10-17　触发了 ajaxSuccess 和 ajaxComplete 事件

10.4　本章小结

本章介绍了 jQuery，这是一种非常流行的开源客户端 JavaScript 框架，可以用来与文档对象模型进行交互。首先介绍了 jQuery 的特点以及在 Web 站点中引入 jQuery 的方法。然后介绍了 jQuery 的基本语法，包括在页面中查找相关元素的 jQuery 选择器和筛选器、对匹配集应用 CSS 和动画效果、添加事件处理、访问通过选择器和筛选器得到的 jQuery 对象以及对 jQuery 对文档的处理方法。在本章的最后，介绍了 jQuery 的扩展应用，包括

下载和使用 jQuery 插件，自己编写插件，以及 jQuery 对 Ajax 的支持等内容。

jQuery 是一个功能强大的工具，通过本章的学习，读者应掌握 jQuery 的基本语法，能够通过 jQuery 简化传统的 JavaScript 代码，为自己的网站开发带来便利。

10.5　思考和练习

1. 在 jQuery 中，使用什么作为在页面中查找元素的快捷方式？
2. 所有的 jQuery 选择器都返回什么？
3. jQuery 的层级选择器有哪些？分别有什么功能。
4. bind()方法和 one()方法有什么不同之处？
5. size()和 length 有何异同点？
6. 有如下<div>元素和<p>元素：

```
<p>卑微的爱</p><div id="box">不如高傲的离开</div>
```

如何使用 jQuery 代码颠倒两个元素的排列顺序。

7. slideUp 和 slideDown 有什么区别？两种方法都接受什么重要参数？
8. jQuery.get()方法与 jQuery.post()方法有何区别？
9. 在使用 load()方法时，所有页面的字符编码应该设置为什么方式？
10. 上机练习，使用 jQuery 实现超链接文件类型标识图标，对于指向不同文件类型的超链接显示不同的标识图标，如图 10-18 所示。



图 10-18　超链接文件类型标识图标

11. 下载 jQuery 插件，根据示例文件，自己编写测试页面，使用插件并查看效果。

第11章　Web站点的发布与部署

本章将主要介绍 Web 应用程序的部署，包括复制 Web 站点、在 IIS 下运行站点和将数据库移动到远程服务器。为了让 Internet 上的用户能够访问 Web 站点，需要将它发布到与 Internet 相连的生产服务器上。通过本章的学习，应重点掌握 Web 站点的部署方法。

本章学习目标：

- 复制 Web 站点
- 在 IIS 下运行 Web 站点
- 将数据移到远程服务器

11.1　部署 Web 站点

网站或 Web 应用程序设计开发完成后，需要发布才能让用户访问。使用什么类型的服务器以及将它定位在哪里的服务器，这取决于具体的需求和预算。可以将站点驻留在阁楼里具有私有 Internet 连接的家用服务器上，或者可以使用能够直接连接到 Internet 主干的外部(通常是商业)提供商的服务器来驻留它。VWD 2010 提供了发布网站的功能，该功能将网站编译为一组可以通过 IIS 直接执行的文件，然后将这些文件复制到目标 Web 服务器上。

11.1.1　部署前的准备工作

当在开发环境中实现 Web 站点的第一个版本时，管理站点及其源代码就非常简单。只有站点源代码的一个版本，因此维护非常容易。然而，一旦将站点投入到生产环境中，就拥有站点的两个版本，一个在生产环境中运行，另一个用于开发。这就很难保持同步。例如，在生产环境中可能使用不同的数据库和连接字符串。如果激活站点时在代码中直接进行所有修改，那么在下一次更新过程中很可能会重写某些设置，从而产生不希望看到的结果。

那么，如何管理相同 Web 站点的不同版本呢？一种比较简单的方法就是将某些硬编码的设置移动到 web.config 文件中。web.config 配置文件使得在开发环境和生产环境中进行不同设置变得很容易。本书前面已经多次使用 web.config 文件来存储与连接字符串有关的信息，还介绍了在<appSettings>元素中通过<add>标记添加用户自定义变量。

例如，在 web.config 中有如下定义：

```
<appSettings>
    <add key="CopyRight" value="版权所有小石头网站" />
</appSettings>
```

在程序中可以使用 System.Web.Configuration 命名空间的 WebConfigurationManager 类来获取该变量的值，从而只使用一行代码就能从这些部分检索数据。

另外，还可以使用表达式语法来访问<appSettings>元素中的数据，格式如下：

```
<%$ AppSettings:AppSettingKeyName %>
```

其中，AppSettingKeyName 表示在 web.config 文件中定义的键。例如，在页面上添加一个 Literal 控件用来显示网站的版权信息，则可以这样设置 Literal 控件的 Text 属性：

```
<asp:Literal ID="Copyright" runat="server" Text="<%$ AppSettings:CopyRight %>" />
```

通常将生产环境和开发环境中参数不一样的值保存在 web.config 文件中，通过上述方法访问这些变量的值，从而使站点变得更容易维护和部署。

将硬编码的应用程序设置移动到 web.config 文件之后，部署过程的下一步就是创建 Web 站点的副本。

11.1.2　复制 Web 站点

在开发站点的过程中，使用 VWD 配置的内置 Web 服务器。虽然这个服务器对于本地开发非常好，但在生产环境中就不能使用它。要将站点投入到生产环境中使用，需要将它部署到运行 IIS 的计算机中——IIS(Internet Information Services)是 Microsoft 的专业 Web 服务器。

要将站点部署到生产服务器中，可以使用表 11-1 显示的部署目标(来自 VWD 内部)。

表 11-1　部署目标

部 署 选 项	描　　述
文件系统	该选项允许在开发计算机或网络化计算机的本地文件系统上创建站点副本。如果稍后要将这些文件手动移动到生产服务器中，那么这个选项就很有用
本地 IIS	该选项允许创建将在本地 IIS 安装下运行的站点的副本
FTP 站点	该选项允许使用 FTP 将组成 Web 应用程序的文件使用发送到远程服务器中
远程站点	该选项允许将组成 Web 应用程序的文件发送到远程 IIS 服务器中。要使这个选项生效，远程服务器需要安装 Front Page Server Extensions 查看 IIS 附带的文档或者咨询远程服务器的管理员可以获得使用这个选项的帮助

如果使用的是 Visual Studio 的商业版本，可以使用 VWD 提供的两种主要的部署方法来访问这 4 个部署选项，复制网站和发布网站。如果使用的是免费的 Express 版本，则只能使用复制网站。本书只介绍复制网站。

复制网站命令可以使用 4 个传输选项中的任意一项来创建站点的副本。这是将站点快速复制到其他位置的好方法。

例 11-1：使用网站复制命令将网站 Chapter7 复制到本地 IIS。

(1) 启动 VWD2010，选择“文件”|“打开网站”命令，打开网站 Chapter 7。

本例在开发阶段已经将发送邮件的方式和邮件地址配置到 web.config 文件中了，所以没有其他需要移到 web.config 文件中的参数了。接下来，需要设置一下站点的起始页。

(2) 选择“网站”|“启动选项”命令，打开站点的属性页，在左侧列表中选择“启动选项”，然后从右侧区域，选择“特定页”单选按钮，单击右侧的“浏览”按钮，打开“选择页码以开始”对话框，这里选择 Index.aspx 页面为起始页，如图 11-1 所示。

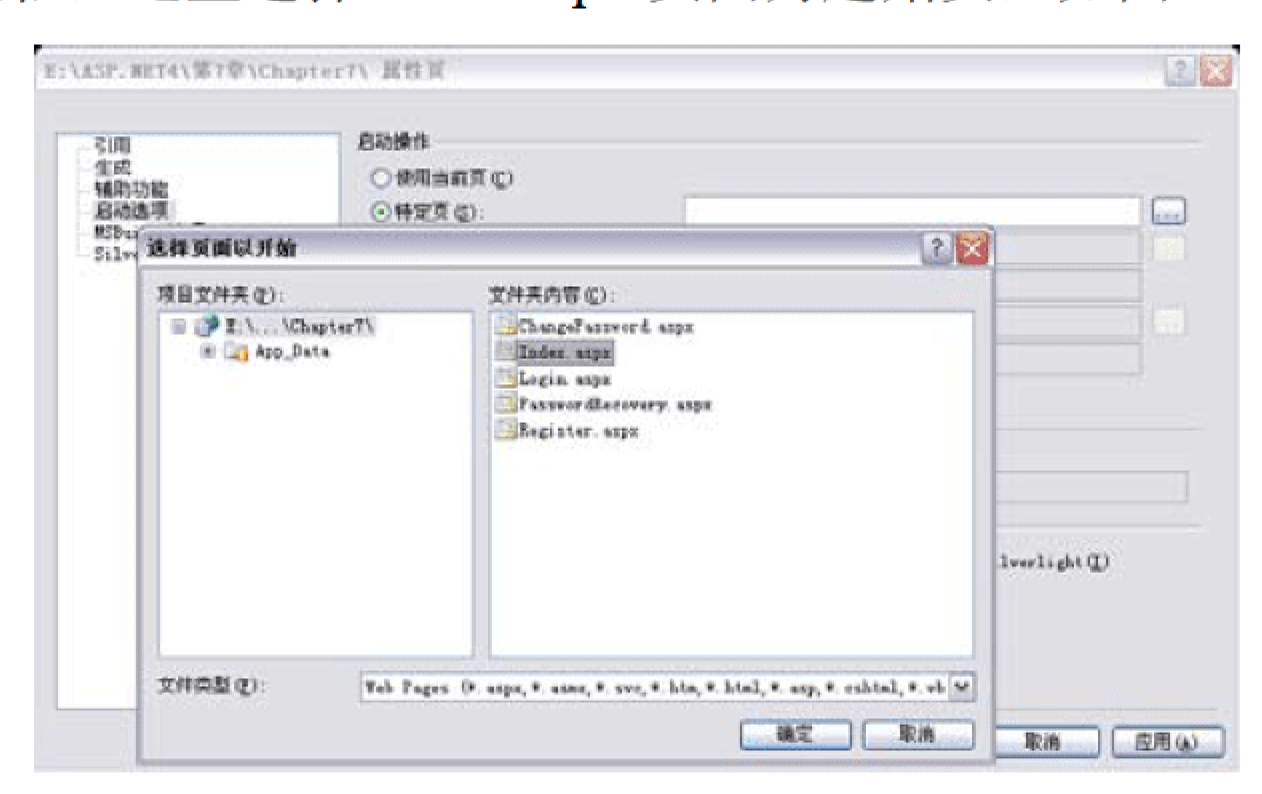


图 11-1　设置站点的起始页

(3) 单击“确定”按钮，完成起始页的设置。

(4) 选择“网站”|“复制网站”命令，打开“复制网站”窗口，如图 11-2 所示。

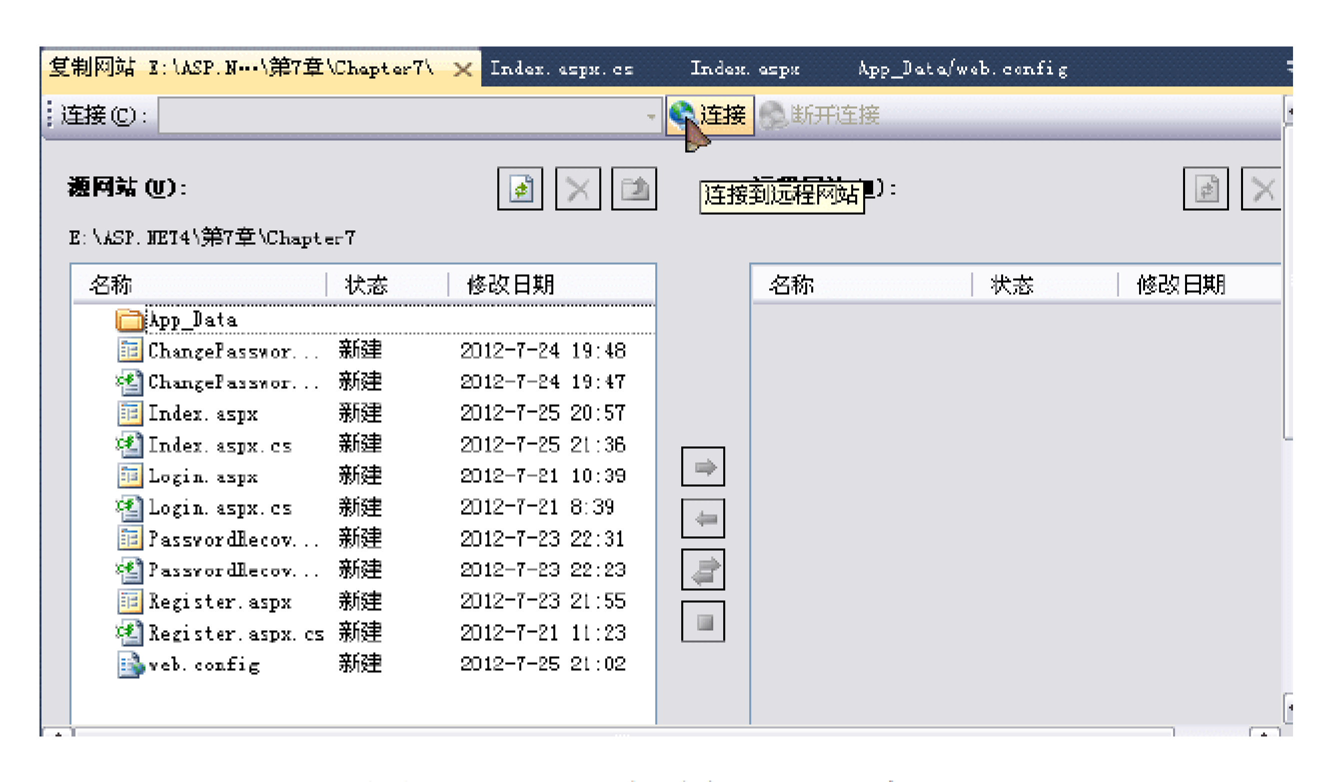


图 11-2　“复制网站”窗口

技巧：

也可以在“解决方案资源管理器”窗口的工具栏中单击复制网站图标打开“复制网站”窗口。

(5) 单击“连接”图标按钮，打开“打开网站”对话框，如图 11-3 所示。在该对话框中选择“本地 IIS”选项，单击右上角的“新建网站”图标新建一个网站，也可以选择一个现有的网站，然后单击“新建虚拟目录”图标为要发布的网站创建一个虚拟目录。

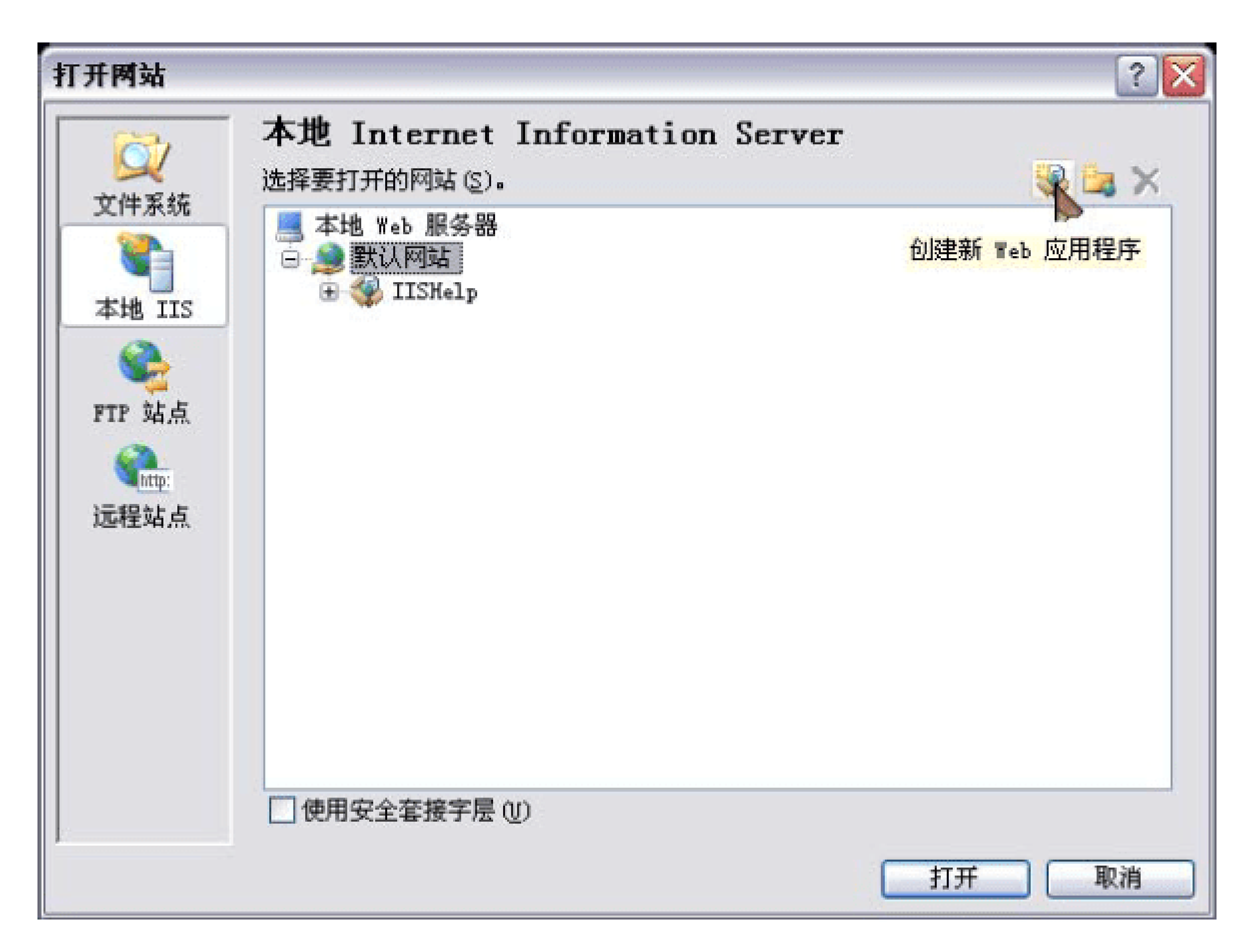


图 11-3　“打开网站”对话框

(6) 创建了网站或虚拟目录后，单击“打开”按钮返回“复制网站”对话框。

(7) 在“源网站”中选择文件，然后单击中间的图标，即可将选中的文件复制到指定的本地 IIS 站点中。

单击“复制网站”窗口中的两个列表之间的“同步选择的文件”按钮，将启动同步进程，在“源网站”和“远程网站”之间同步所有文件。

11.2　在 IIS 下运行站点

为了在 IIS 下运行 Web 站点，需要执行下面几个步骤：

(1) 安装和配置 IIS；

(2) 安装和配置.NET Framework 4；

(3) 配置安全设置。

根据系统的当前状态，有些操作是可选的。下面将介绍如何实现这些步骤。

11.2.1　安装和配置 Web 服务器

虽然大多数 Windows 版本都包含 IIS，但默认情况下不会安装它，因此首先要确保使用的 Windows 版本支持 IIS，然后就是安装 IIS。虽然 Windows Vista 和 Windows 7 的 Starter 版本和 Home Basic 版本提供了部分 IIS，但不能在它们上面运行 ASP.NET 页面，因此至少需要安装 Home Premium 版本。Windows 基于服务器的版本则完全支持 IIS。

要在 Windows 上安装和配置 IIS，需要以 Administrator(管理员)身份登录系统。除了安装 IIS 之外，还要知道如何在 IIS 中创建和配置 Web 站点。

1. 安装 IIS

本节将介绍如何安装 IIS，不同版本的安装过程略有不同，下面以 Windows XP 和 Windows 7 为例介绍安装的步骤。

在 Windows XP 和 Windows Server 2003 系统中，可以通过“控制面板”中的“添加或删除程序”来安装 IIS，或者通过选择“开始”|“运行”命令，然后通过输入 appwiz.cpl 来打开“添加和删除程序”对话框。接着，单击对话框左边的“添加/删除 Windows 组件”图标，打开“Windows 组件向导”对话框，如图 11-4 所示。

在“组件”列表中，选中“Internet 信息服务(IIS)”复选框，然后单击“详细信息”按钮。在打开的对话框中至少选中“公用文件”和“Internet 信息服务管理单元”复选框，其他选项为可选。

在 Windows 7 和 Windows Vista 系统中，通过“程序和功能”部分来安装 IIS，可以通过“控制面板”或单击“开始”按钮，在“搜索”框中输入 appwiz.cpl，然后按回车键来访问这个部分。在“程序和功能”中，单击“打开和关闭 Windows 功能”链接来打开“Windows 功能”对话框，如图 11-5 所示。

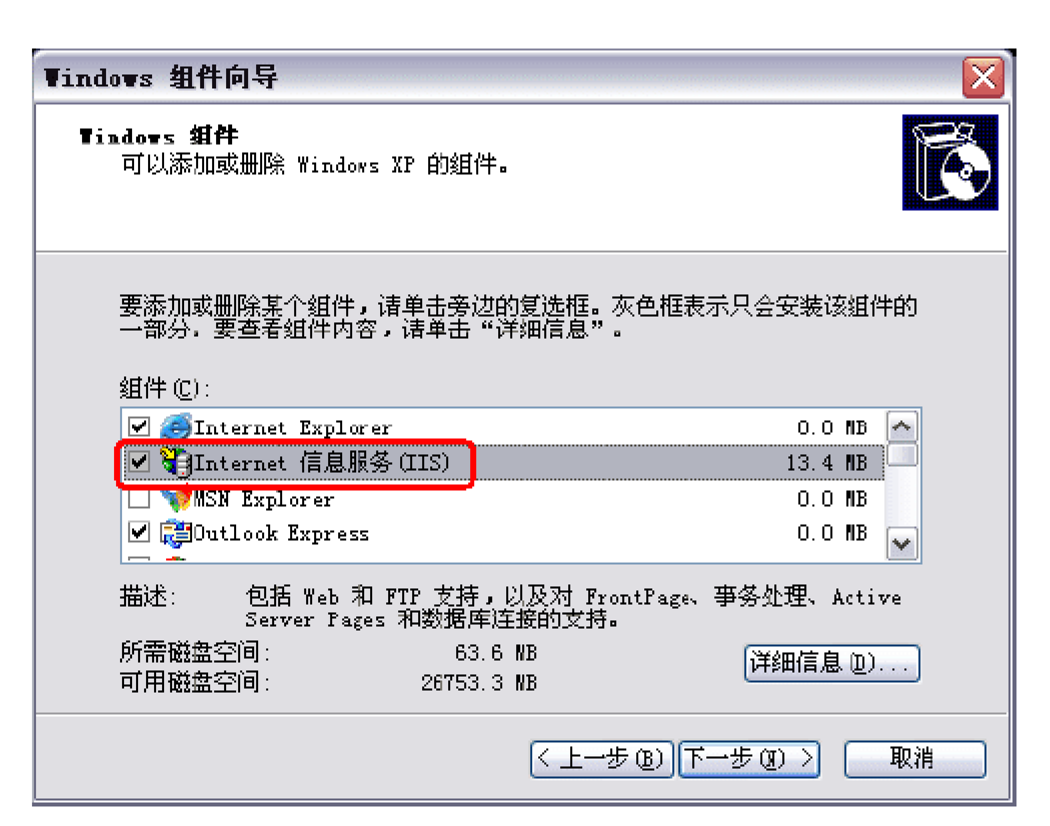


图 11-4　“Windows 组件向导”对话框

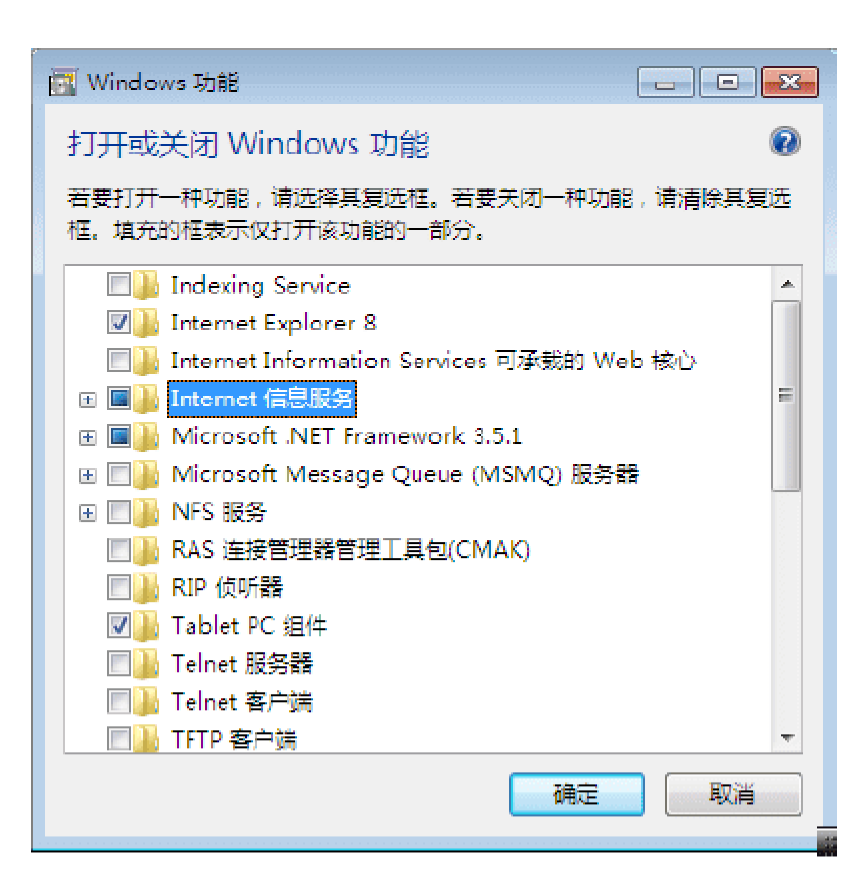


图 11-5　“Windows 功能”对话框

单击“Internet 信息服务”选项，这会选择它的一些必需的子功能。然后展开“Internet 信息服务”|“万维网服务”|“应用程序开发功能”，并选择 ASP.NET 选项。这也会选中其他一些相关的功能。最后，单击“确定”按钮，Windows 将开始安装所选的功能。

说明：

IIS 安装过程中，会提示用户插入系统盘，请事先准备操作系统的安装介质。

2. 安装和配置 ASP.NET

成功安装 IIS 之后，还要确保已经安装了 Microsoft .NET Framework 4。如果在目标计算机上安装了 VWD 2010，那么就已经安装了.NET Framework 4。否则就要从 Microsoft 站点中下载，其地址是 http://msdn.microsoft.com/en-us/netframework。下载之后，可以运行安装程序，并按照向导提示操作。

如果计算机上已经安装了.NET Framework 4，后来才安装 IIS，那么就要告诉 IIS 已经存在 Framework。通常情况下，这将在.NET Framework 的安装过程中完成。如果后来才安装 IIS，那么就要手动完成该操作。

在 IIS 中注册 ASP.NET 的步骤如下。

(1) 打开命令行窗口。

(2) 通过输入下面的命令导航到.NET Framework 4 文件夹：

```
cd   \WINDOWS\Microsoft.NET\Framework\v4.0.30319
```

注意：

在用户的计算机上，v4.0 后欧盟的实际版本号可能会有所不同。另外，如果使用的是 64 位的 Windows 版本，那么 Framework 文件夹命名则为 Framework64。

(3) 输入 aspnet_regiis –i，之后会收到 ASP.NET 4 已经成功地安装在 IIS 中注册的消息。

11.2.2 IIS 中的安全性

由于 VWD 2010 中内置 Web 服务器的无缝集成，可能还不知道内部发生的情况，也不知道在浏览站点中的页面时哪些安全设置有效。为了使用站点内的资源，例如 ASPX 文件、后台代码文件、App_Data 文件夹中的数据库和站点内的图像，Web 服务器需要从 Windows 获得访问这些资源的权限。这就意味着要配置 Windows，授权 Web 服务器使用的账户访问这些资源的权限。

需要权限的特定账户取决于 Windows 版本，以及是在 IIS 下运行站点还是使用内置 Web 服务器运行站点。

内置 Web 服务器使用的账户是用来登录 Windows 计算机的账户。这个账户通常是“域名\用户名”或“机器名\用户名”。在 Windows 上使用这个账户登录，就启动了 VWD 2010，它再启动内置的 Web 服务器，整个 Web 服务器都使用证书凭据运行。由于通常登录计算机的账户是本地 Windows 计算机上的 Administrator 或者超级用户，具有访问组成站点的所有文件的权限，因此到目前为止可能一切正常，不需要修改安全设置。

如果使用的是 IIS，情况则完全不同。在默认情况下，IIS 下的 ASP.NET 应用程序使用在安装 IIS 时创建的特定账户运行。在 Windows XP 中，这个账户名为 ASPNET；在 Windows Vista 以及 Windows Server 2003 和 2008 中，这个账户名为 Network Service；而在 Windows 7 和 Windows Server 2008 R2 中则为 ApplicationPoolIdentity。除了 ASP.NET 应用程序使用的账户之外，还需要配置 Web 服务器用于访问资源的账户，这些资源不直接与 ASP.NET 相关，如图像、CSS 文件等。

注意：

在系统中不能直接找到 ApplicationPoolIdentity 用户账户，因为它取决于配置的应用程序池的名称。

在找到需要配置的账户之后，最后一步就是配置文件系统。

不管使用的是哪个账户，都需要修改 Windows 文件系统，从而允许 Web 服务器访问资源。这只有在使用 NTFS 而不是 FAT 或 FAT32(旧的 Microsoft 文件系统)格式化硬盘驱动器时才有必要。

在标准 Windows 系统上，都使用 Windows NTFS 文件系统保护所有文件和文件夹。为了确保 Web 站点正确运行，需要给 Web 服务器使用的账户授予必要的权限，以允许他们访问 Web 站点上的文件和文件夹。对于大多数文件和文件夹而言，具有读取权限就足够了。然而，对于 App_Data 文件夹通常需要在运行时写入，因此需要给账户授予对这个文件夹的修改和写入权限。

例 11-2：以 App_Data 文件夹为例，授予 ASPNET 用户对文件夹的访问权限。

(1) 打开 Windows 资源管理器，找到 App_Data 文件夹，右击该文件夹，从弹出的快捷菜单中选择“属性”命令，打开该文件夹的属性对话框，选择“安全”选项卡，如图 11-6 所示。此时在“组或用户名称”列表中还没有 ASPNET 用户。

(2) 单击“添加”按钮，打开“选择用户或组”对话框，在“输入对象名称来选择”文本区域中输入 ASPNET，然后单击“检查名称”按钮，系统将自动检查，并以“机器名/用户名”的形式更新“输入对象名称来选择”文本区域，如图 11-7 所示。



图 11-6　文件夹属性对话框的“安全”选项卡



图 11-7　“选择用户或组”对话框

技巧：

也可以单击“选择用户或组”对话框中的“高级”按钮进行查找，然后从用户列表中选择 ASPNET 用户。

(3) 单击“确定”按钮，关闭“选择用户或组”对话框，此时，ASPNET 用户将被添加到“组合用户名称”列表中。默认情况下，该用户对当前文件夹有“读取和运行”、“列出文件夹目录”和“读取”的权限，如图 11-8 所示。

(4) 只需选择“写入”权限对应的“允许”列的复选框即可为 ASPNET 用户增加写入权限，单击“应用”按钮，使设置生效。

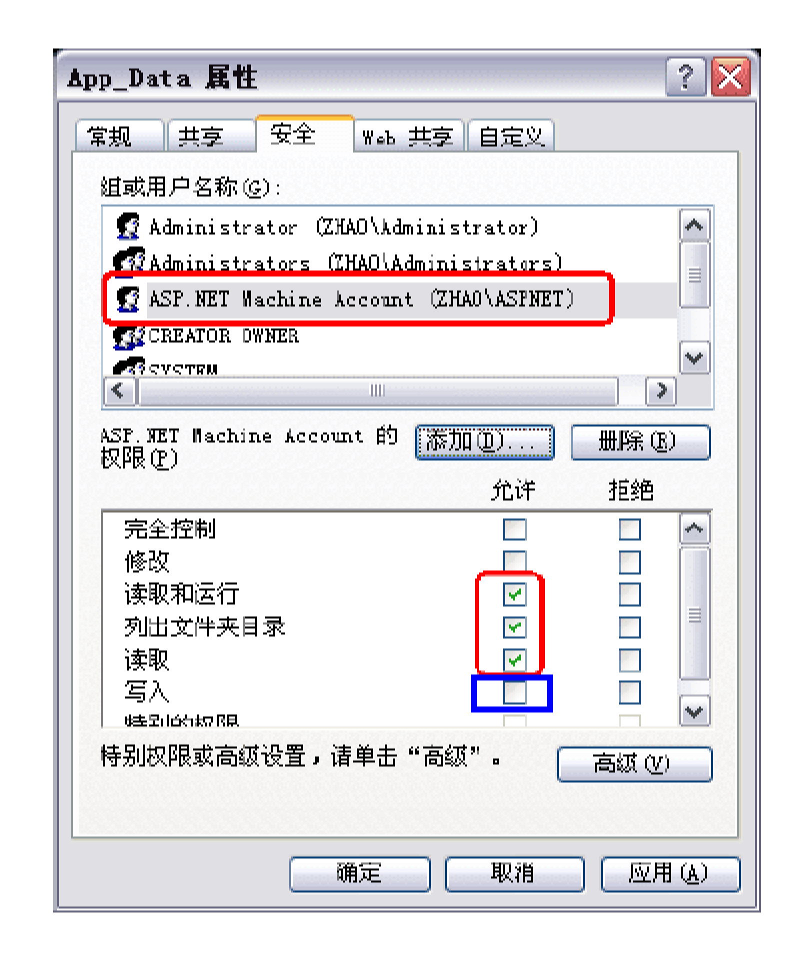


图 11-8　ASPNET 用户的默认权限

11.3　将数据移到远程服务器

将站点发布到本地计算机上的 IIS 中非常简单。只要将数据复制到新位置，配置 IIS，然后修改一些安全设置即可。因为站点继续使用 SQL Server 2008 Express 版本，它会正常运行。

如果要将站点移动到外部服务器或主机，事情就没那么简单了。虽然使用 FTP 复制组成站点的文件非常简单，但将数据从 SQL Server 2008 数据库复制到另一台主机通常有些诀窍。这是因为大多数 Web 主机不支持 SQL Server 2008 Express 版本，因此不能只将.mdf 文件复制到远程主机上的 App_Data 文件夹中。相反，这些主机通常提供 SQL Server 的完全版本，可以使用基于 Web 的管理工具或使用像 SQL Server Management Studio 这样的工具来访问它们。

11.3.1　使用 Database Publishing Wizard

为了方便地将数据从本地 SQL Server 2008 数据库传送到 Web 主机的 SQL Server 数据库中，Microsoft 创建了 Database Publishing Wizard。

Database Publishing Wizard 允许创建.sql 脚本，它包含在远程服务器上重建数据库及其数据所需的全部信息和远程服务器上的数据。重建数据的步骤如下。

(1) 从本地 SQL Server 数据库创建.sql 脚本；

(2) 将这个脚本发送到远程主机并执行。

例 11-3：导出 WeiBo 数据库。

(1)　在 VWD 中打开任意一个项目，选择“视图”|“服务器资源管理器”命令打开“服务器资源管理器”窗口。

(2) 展开“数据连接”节点，右击 WeiBo.dbo 数据库，从弹出的快捷菜单中选择 Publish to Provider 命令，打开 Database Publishing Wizard 对话框。首先弹出的是欢迎页面，如图

11-9 所示。

(3) 单击“下一步”按钮，进入“选择数据库”页面，如图 11-10 所示。



图 11-9　欢迎页面



图 11-10　“选择数据库”页面

(4) 选择 WeiBo 数据库，并选中“为所选数据库中的所有对象编写脚本”复选框，然后单击“下一步”按钮。进入“选择输出位置”页面，如图 11-11 所示。

(5) 在这个页面中，有两个选项。第一个选项允许使用所需的 SQL 语句创建文本文件，第二个选项允许通过 Internet 直接与共享的主机提供商会话。此处选择“将脚本保存到文件”单选按钮，在“文件名”文本框中输入保存的位置和文件名，单击“下一步”按钮，进入“选择发布选项”页面，如图 11-12 所示。



图 11-11　“选择输出位置”页面



图 11-12　“选择发布选项”页面

(6) 继续单击“下一步”按钮，进入“检查摘要”页面，如图 11-13 所示，该页显示了前面所做的选择。

(7) 单击“完成”按钮，将出现“数据库发布进度”页面，如图 11-14 所示。向导会在指定文件夹中生成 SQL 脚本。单击“关闭”按钮，完成数据库导出操作。



图 11-13　“检查摘要”页面

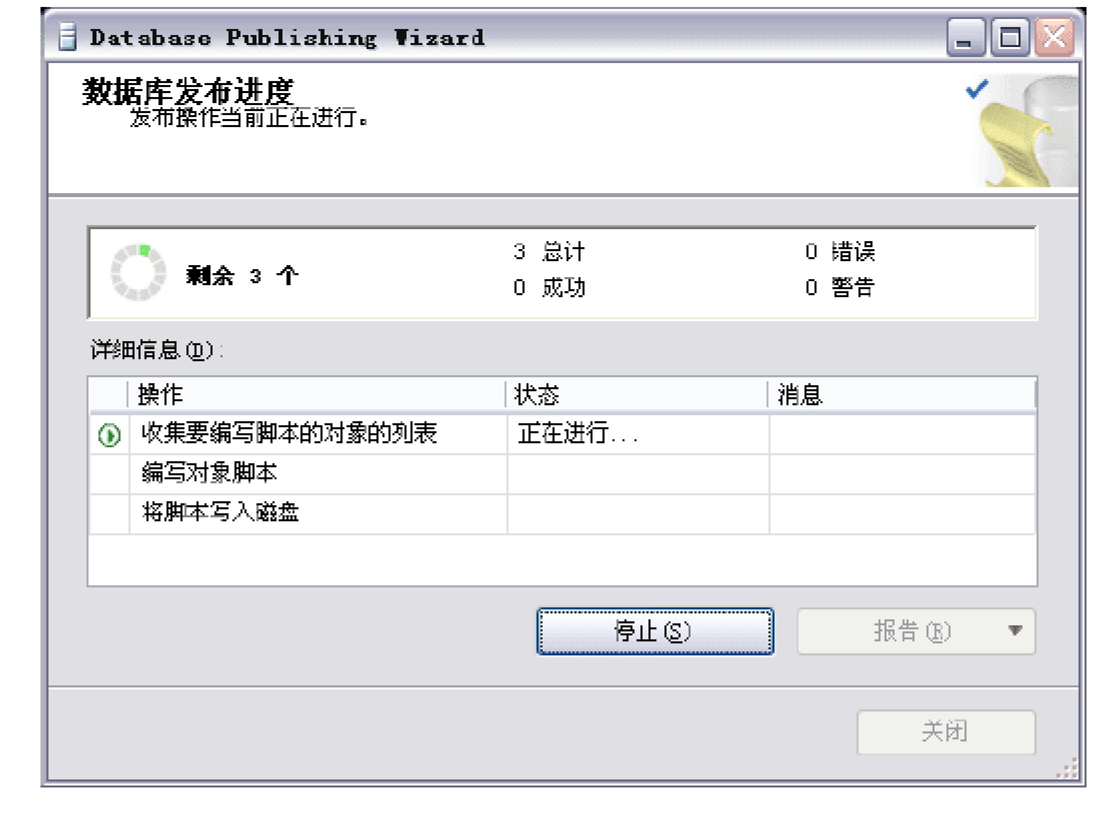


图 11-14　“数据库发布进度”页面

(8) 可以用记事本打开生成的.sql 文件，查看它包含的 SQL 语句。且可以使用它在兼容的 SQL Server 2008 数据库上重建数据库。

数据库的内容可以分为两类：数据库的结构和实际数据。Database Publishing Wizard 运行时，首先调查数据库的结构，并为它在数据库内找到的所有项创建 SQL CREATE 语句，然后创建 INSERT 语句，这些语句在目标数据库内重建所有记录。

11.3.2　创建数据库

虽然每个主机在提供对 SQL Server 的访问权时都有自己的规则和程序，但它们可以分为 3 类。

- 第一类，有些主机不允许远程访问数据库，它要求提交.sql 文件以便执行它。在这种情况下，除了发送文件然后等待主机创建数据库之外，不需要做任何事情。
- 第二类，包含的主机允许通过 Web 接口执行 SQL 语句。通常需要登录联机控制面板，然后通过上传文件或者将其内容粘贴到 Web 页面中的文本区，来执行 Database Publishing Wizard 导出的 SQL 语句。不管使用哪种方法，最终都要使用从应用程序可以访问的数据库。
- 第三类，包含的主机允许通过 Internet 连接到 SQL Server。这允许使用像 SQL Server Management Studio 这样的工具从桌面连接到主机上的数据库，并远程执行 SQL 脚本。

SQL Server Management Studio 也有免费的 Express 版本，可以从 Microsoft 网站(http://www.microsoft.com/vstudio/express/sql)下载。该工具的运行原理与商业版几乎相同。

在目标服务器上重建数据库之后，需要修改 Web 站点内的连接字符串，以便重新配置 ASP.NET 应用程序，从而使用新的数据库。需要修改两个连接字符串：一个是连接用户数据库的连接字符串，另一个是 ASP.NET Application Services 默认使用的 LocalSqlServer 连接字符串。

11.4　本章小结

部署是 Web 站点开发后期的重要操作。然而，不可能只部署站点一次。只要发布了站点的第一个版本，就要考虑添加其他的新功能，因此应尽量让站点部署比较容易且灵活。通常在部署之前，需要将硬编码的配置设置移动到 web.config 文件中，然后使用 Database Publishing Wizard 导出数据库，以便在远程服务器重建数据库，最后通过“复制站点”或“发布站点”功能实现站点的复制和发布。通过本章的学习，读者应掌握站点的部署与发布的基本操作，并能够将网站所需的数据库迁移到远程服务器上。

11.5　思考和练习

1. 要将站点部署到生产服务器中，可以使用哪些部署目标？
2. 运用本章所学知识，将前面创建的网站复制到本地 IIS。
3. 简述将数据库导出为 sql 脚本的过程和步骤。

第12章　简易微博系统

微博，即微型博客，是随着 Web 2.0 而兴起的一类开放的互联网社交服务，它允许用户以简短文字随时随地更新自己的状态，每条信息的长度都在 140 字以内，支持图片、音频和视频等多媒体的出版，每个用户既是微内容的创造者也是微内容的传播者和分享者。

本章将综合运用前面所学内容，创建一个简易的微博平台，支持用户发表微博，评论或转播微博，以及用户之间的收听等功能。通过本章的学习，读者将掌握一个独立的 Web 站点从设计到实现的开发流程和基本方法，同时复习全书所学内容。

本章学习目标：

- 进一步熟悉 ASP.NET 编程技术
- 掌握网站设计与实现的基本流程

通用类的创建

- 业务逻辑与界面展示的分离
- 使用母版页
- 综合运用本书所学知识

12.1　系统设计

一个完整的软件系统开发过程分为软件定义阶段、软件开发阶段和软件运行维护阶段。

- 软件定义阶段主要决定将要开发软件的功能和特性。它又可以细分为问题的定义、可行性研究、需求分析 3 个阶段。
- 软件开发阶段又可细分为总体设计、详细设计、程序编制和软件测试 4 个阶段。
- 软件运行维护阶段的主要任务是通过各种必要的维护活动使系统持久地满足用户的需求。

本章要开发的是一个简易的微博系统，将重点介绍需求分析和总体设计以及编码实现。

12.1.1　需求分析

微博是一个基于用户关系的信息分享、传播以及获取平台。它十分简单，也十分便捷，即时通讯功能非常强大，一般发布的消息只是由简单的只言片语组成，对用户的技术要求门槛很低，而且在语言的编排组织上不需要长篇大论，只需要反映自己的心情即可。

在本微博系统中，用户可以发表、转发或评论消息(由于篇幅所限，本系统只支持文本消息)，随时看到被关注者的最新动态，还可以浏览非关注者发表的最新消息。用户之间可

以收听或者取消收听。用户对自己发表的消息可以进行删除等操作。

系统的用例图如图 12-1 所示。

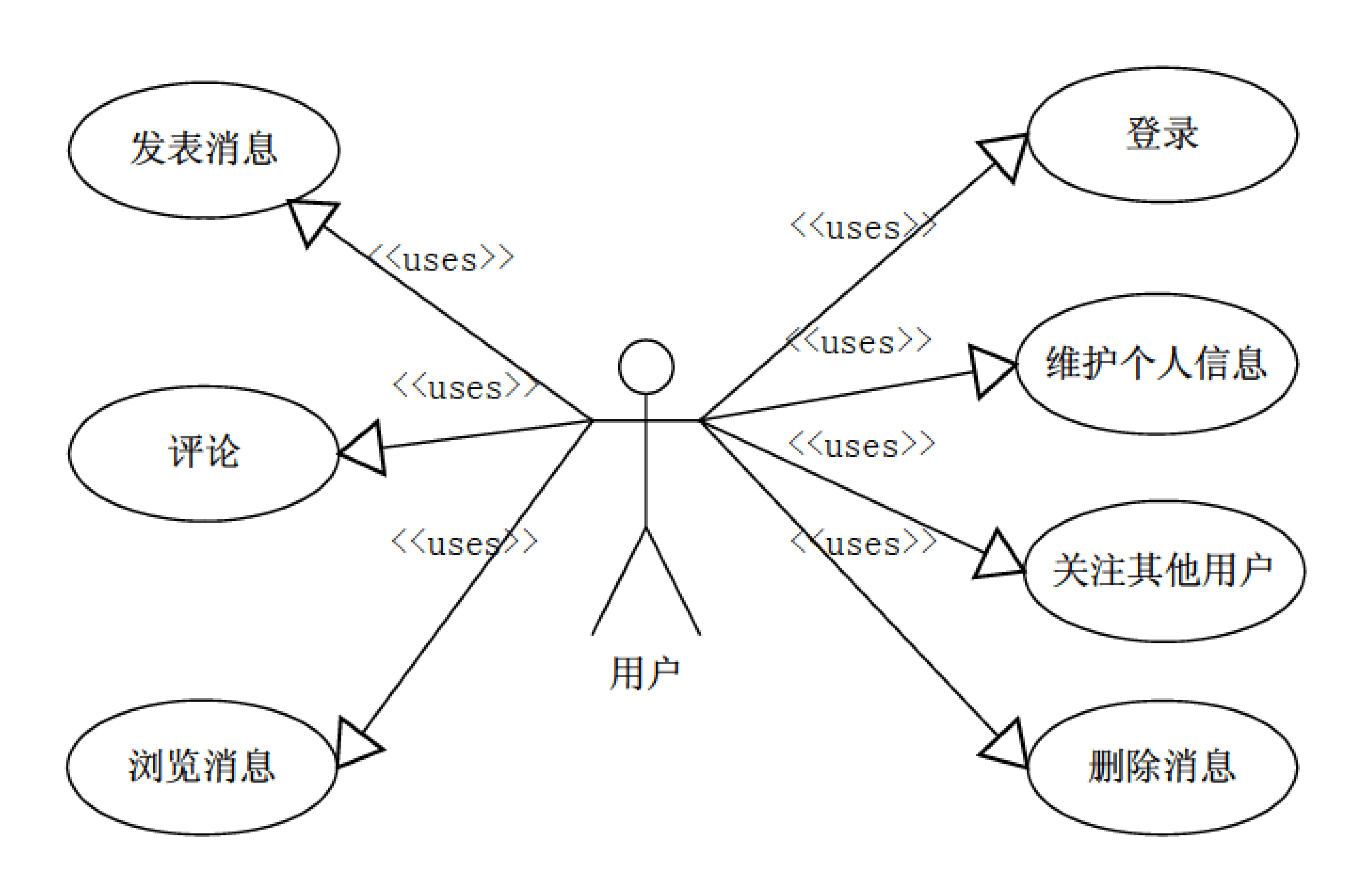


图 12-1　微博系统用例分析

12.1.2　数据库设计

本系统涉及的数据实体有用户、消息和评论。另外还有一个用户与用户之间的关系。相应的数据库设计我们在本书第 5 章已经介绍过，请参考本书 5.1.2 节。

12.2　系统实现

本节将详细介绍微博系统的实现。包括数据操作类的创建与实现、实体类的创建、以及各 Web 窗体的设计与实现。

首先启动 VWD 2010，新建空网站 MiniWeiBo。

12.2.1　数据访问类

因为几乎所有页面都涉及数据库的访问，所有把访问数据库的操作抽象为一个独立的公共类文件 DB.cs。并把数据库连接字符串存放到 web.config 配置文件中。

在 web.config 配置文件中设置数据库连接信息，添加语句如下：

```
        <connectionStrings>
            <add name="WeiBoDb" connectionString="Data Source=zhao\sqlexpress;Initial
Catalog=WeiBo;Integrated Security=True"
                providerName="System.Data.SqlClient" />
        </connectionStrings>
```

在“解决方案资源管理器”窗口中，为项目添加 ASP.NET 文件夹 App_Code，在

App_Code 目录中，添加一个 C#类文件 DB.cs，在该类中，添加对数据库的操作，代码如下：

```
using System.Data;
using System.Data.SqlClient;
using System.Collections;
using System.Web.Configuration;
public class DB
{
    private SqlConnection con = null;
    private string strConn;
    private SqlTransaction tran;
    public DB()
    {
        strConn = WebConfigurationManager.ConnectionStrings["WeiBoDb"].ToString();
    }
    //打开数据库连接
    public void open()
    {
        if (con == null )
            con = new SqlConnection(strConn);
        if (con.State.Equals(ConnectionState.Closed))
            con.Open();
    }
    //关闭数据库
    public void close()
    {
        if (con == null)
            return;
        if (con.State.Equals(ConnectionState.Open))
        {
            con.Close();
            con.Dispose();
            con = null;
        }
        else
        {
            con.Dispose();
            con = null;
        }
    }
    //执行 SQL 语句
    public int ExecuteSQLNonQuery(string sqlStr)
    {
        try
        {
```

```
            this.open();//打开连接
            SqlCommand cmd = new SqlCommand(sqlStr, con);
            return cmd.ExecuteNonQuery();
        }
        catch
        {
            return -1;
        }
        finally
        {
            close();
        }
    }
    //执行 SQL 语句，不关闭连接，事务处理使用
    public int ExecuteSQLNonQueryWithTran(string sqlStr)
    {
        int ret = 0;
        try
        {
            this.open();//打开连接
            SqlCommand cmd = new SqlCommand(sqlStr, con);
            cmd.Transaction = tran;
            ret = cmd.ExecuteNonQuery();
        }
        catch (Exception e)
        {
            throw e;
        }
        return ret;
    }
    //执行 SQL 语句,传递参数给 SqlCommand
    public int ExecuteSQLNonQueryWithParam(string sqlStr, List<SqlParameter> sqlParams)
    {
        try
        {
            this.open();//打开连接
            SqlCommand cmd = new SqlCommand(sqlStr, con);
            foreach(SqlParameter p in sqlParams)
                cmd.Parameters.Add(p);
            return cmd.ExecuteNonQuery();
        }
        catch
        {
            return -1;
```

```
        }
        finally
        {
            close();
        }
    }
    //执行 SQL 语句，返回查询的表
    public DataTable GetDataTable(string sqlStr)
    {
        DataTable dt;
        try
        {
            open();
            SqlDataAdapter sda = new SqlDataAdapter(sqlStr, con);
            DataSet ds = new DataSet();
            sda.Fill(ds);
            dt = ds.Tables[0];
        }
        catch (Exception ex)
        {
            dt = null;
        }
        finally
        {
            close();
        }
        return dt;
    }
    //执行 SQL 语句，返回 DataRow
    public DataRow GetDataRow(string sqlStr)
    {
        DataRow dr;
        try
        {//调用该类 GetDataTable 方法
            dr = GetDataTable(sqlStr).Rows[0];
        }
        catch
        {
            dr = null;
        }
        finally
        {
            close();
        }
```

```
            return dr;
        }
        public void BeginTrans()
        {
            tran = con.BeginTransaction();
        }
        public void Commit()
        {
            tran.Commit();
        }
        public void Rollback()
        {
            tran.Rollback();
        }
    }
```

该类中包括一个构造函数 DB()，打开数据库连接方法 open()，关闭连接方法 close()，执行非查询 SQL 语句方法 ExecuteSQLNonQuery(string sqlStr)，执行带有参数的 SQL 语句方法 ExecuteSQLNonQueryWithParam，获取 DataTable 对象的查询方法 GetDataTable(string sqlStr)和获取 DataRow 的查询方法 GetDataRow(string sqlStr)，以及事务处理相关的几个方法。

12.2.2　数据实体类

在微博系统中的操作可分为两类，一类是针对用户的，另一类是针对消息的。所以要把用户和消息分别封装为不同的实体类，然后在类中实现相应的操作。

1. 用户实体类 Zuser.cs

在 App_Code 目录中添加新的类文件 Zuser.cs。在该类中添加操作 Z_USER 表所需的成员变量。通常类成员变量定义为私有的，然后定义相应的属性来访问这些私有变量，本书为了使程序简化，直接将成员定义为自动属性。相应的代码如下：

```
        public int user_id{get;set;}
        public string user_name{get;set;}
        public string user_login          {get;set;}
        public string user_password       {get;set;}
        public string user_sex            {get;set;}
        public byte[] user_photo          {get;set;}
        public string user_email          {get;set;}
        public DateTime regist_time       {get;set;}
        public string user_address        {get;set;}
        public DateTime user_birthday     {get;set;}
        public string user_telephone      {get;set;}
        public string home_url            {get;set;}
```

```
    public string user_info         {get;set;}
    public string errorMsg{get;set;}     //用于存放错误消息
```

除了 Z_USER 表字段的自动属性之外，上述代码还定义了一个 errorMsg 属性，该属性的作用是当对用户的操作失败或出现异常时，反馈错误信息给调用者。

接下来，添加一个构造函数，根据指定的用户 id 初始化用户信息，代码如下：

```
    public Zuser(int userId)//构造函数
    {
        string sqlStr = "SELECT * FROM Z_USER WHERE user_id=" + userId.ToString();
        DB db = new DB();
        DataRow dr = db.GetDataRow(sqlStr);
        if (dr == null)
        {
            errorMsg = "指定的用户 id 不存在";
            user_id = -1;
        }
        else
        {
            Init(dr);
        }
    }
    private void Init(DataRow dr)//初始化
    {
        user_id = Int32.Parse(dr["user_id"].ToString());
        user_name = dr["user_name"].ToString();
        user_login = dr["user_login"].ToString();
        user_password = dr["user_password"].ToString();
        user_sex = dr["user_sex"].ToString();
        if (dr.IsNull("user_photo"))
            user_photo = null;
        else
            user_photo = (byte[])dr["user_photo"];
        if (dr.IsNull("user_email"))
            user_email = null;
        else
            user_email = dr["user_email"].ToString();
        regist_time = (DateTime)dr["regist_time"];
        if (dr.IsNull("user_address"))
            user_address = null;
        else
            user_address = dr["user_address"].ToString();
        if (dr.IsNull("user_birthday"))
            ;
        else
```

```
                user_birthday = (DateTime)dr["user_birthday"];
            if (dr.IsNull("user_telephone"))
                user_telephone = null;
            else
                user_telephone = dr["user_telephone"].ToString();
            home_url = dr["home_url"].ToString();
            if (dr.IsNull("user_info"))
                user_info = null;
            else
                user_info = dr["user_info"].ToString();
        }
```

与用户相关的操作主要有如下几个。

- 注册新用户，包括注册前的检查用户名和个人微博地址是否可用。
- 用户登录系统，登录成功后获取用户收听的用户数、用户的听众数和用户发表的消息数等信息。
- 个人信息维护。
- 发表新消息。
- 收听和取消收听其他用户，包括检查是否已经收听某用户。
- 查找用户。

相应方法的实现如下所示：

```
        public bool login(string userLogin, string userPassword)//登录
        {
            string sqlStr = "SELECT * FROM Z_USER WHERE user_login='" + userLogin +"'";
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            if (dr == null)
            {
                errorMsg = "登录名不存在";
                return false;
            }
            if (dr["user_password"].ToString() == userPassword )
            {
                Init(dr);
                return true;
            }
            else
            {
                errorMsg = "密码输入有误";
                return false;
            }
        }
        public bool register()//注册
```

```
{
    bool ret = true;
    string sqlString = "INSERT INTO Z_USER(user_name,user_login," +
        "user_password,user_sex,home_url) values('" + user_name + "','" +
        user_login + "','" + user_password+ "','" + user_sex + "','" + home_url + "')";
    DB db = new DB();
    db.open();
    db.BeginTrans();
    try
    {
        int result = db.ExecuteSQLNonQueryWithTran(sqlString);
        if (result < 1)
        {
            db.Rollback();
            throw new Exception("插入新记录失败");
        }
        sqlString = "UPDATE Z_USER SET user_name = user_name";
        if (user_email.Length > 0)
            sqlString += ",user_email='" + user_email + "'";
        if (user_info.Length > 0)
            sqlString += ",user_info='" + user_info + "'";
        if (user_address.Length > 0)
            sqlString += ",user_address='" + user_address + "'";
        sqlString += " WHERE user_login='" + user_login + "'";
        result = db.ExecuteSQLNonQueryWithTran (sqlString);
        if (result < 1)
        {
            db.Rollback();
            throw new Exception("操作失败");
        }
        db.Commit();
    }
    catch (Exception e)
    {
        db.Rollback();
        ret = false;
        errorMsg = e.Message;
    }
    finally
    {
        db.close();
    }
    return ret;
}
public bool checkLogin(string userLogin)//检查登录名是否可用
{
    string sqlStr = "SELECT * FROM Z_USER WHERE user_login='" + userLogin + "'";
```

```
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            if (dr == null)
                return false;//用户名不存在
            return true;
        }
        public bool checkUrl(string homeUrl)//检查个人微博地址是否可用
        {
            string sqlStr = "SELECT * FROM Z_USER WHERE home_url='" + homeUrl + "'";
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            if (dr == null)
                return false;//个人微博地址不存在
            return true;
        }
        public string getOther(int userId)//获取用户收听的用户数
        {
            string sqlStr = "SELECT COUNT(*) FROM Z_USER_FUN WHERE user_id = " +
user_id.ToString();
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            return dr[0].ToString();
        }
        public string getFun(int userId)//获取用户的听众数
        {
            string sqlStr = "SELECT COUNT(*) FROM Z_USER_FUN WHERE fun_user_id = " +
user_id.ToString();
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            return dr[0].ToString();
        }
        public string getMsg(int userId)//获取用户发表的消息数
        {
            string sqlStr = "SELECT COUNT(*) FROM Z_MESSAGE WHERE user_id = " +
user_id.ToString();
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            return dr[0].ToString();
        }
        public byte[] getPhotoById(string userId)//根据 user_id 获取用户头像，用于显示
        {
            string sqlStr = "SELECT user_sex,user_photo FROM Z_USER WHERE user_id = " + userId;
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            user_sex = dr["user_sex"].ToString();//返回性别用于设置默认头像(对于为上传头像的用户)
            if(dr.IsNull("user_photo"))
                return null;
            else
```

```
                return (byte[])dr["user_photo"];
        }
        public DataTable getUserOther(string userId)//获取用户收听的用户列表
        {
            string sqlStr = "SELECT *,'touxiang.aspx?userid=' + LTRIM(STR(user_id)) AS photo_path " +
                " FROM Z_USER WHERE user_id IN (SELECT fun_user_id FROM Z_USER_FUN
WHERE user_id = " + userId + ")";
            DB db = new DB();
            return db.GetDataTable(sqlStr);
        }
        public DataTable getUserFun(string userId)//获取用户的听众列表
        {
            string sqlStr = "SELECT *,'touxiang.aspx?userid=' + LTRIM(STR(user_id)) AS photo_path " +
                " FROM Z_USER WHERE user_id IN (SELECT user_id FROM Z_USER_FUN
WHERE fun_user_id = " + userId + ")";
            DB db = new DB();
            return db.GetDataTable(sqlStr);
        }
        public DataTable searchUser(string name,string address)//模糊查询指定条件的用户
        {
            string sqlStr = "SELECT *,'touxiang.aspx?userid=' + LTRIM(STR(user_id)) AS photo_path
" +
                " FROM Z_USER WHERE user_name LIKE '%" + name + "%'";
            if(address.Length>0)
                sqlStr += " AND user_address LIKE '%" + address + "%'";
            DB db = new DB();
            return db.GetDataTable(sqlStr);
        }
        public bool hasListen(string uid)//是否已收听指定用户
        {
            string sqlStr = "SELECT * FROM Z_USER_FUN WHERE user_id ="+user_id.ToString()
+ " AND fun_user_id="+uid;
            DB db = new DB();
            return (db.GetDataRow(sqlStr)!=null);
        }
        public void listen(string uid)//收听某个用户
        {
            string sqlStr = "INSERT INTO Z_USER_FUN(user_id,fun_user_id) VALUES(" +
user_id.ToString() + "," + uid+")";
            DB db = new DB();
            db.ExecuteSQLNonQuery(sqlStr);
        }
        public void cancelListen(string uid)//取消收听某个用户
        {
```

```
                string sqlStr = "DELETE FROM Z_USER_FUN WHERE user_id =" + user_id.ToString()
+ " AND fun_user_id=" + uid;
                DB db = new DB();
                db.ExecuteSQLNonQuery(sqlStr);
            }
            public void postMsg(string strMsg)//发表新消息
            {
                string sqlStr = "INSERT INTO Z_MESSAGE(user_id,msg_content,reply_count,post_time)
VALUES(" +
                    user_id.ToString() + ",'" + strMsg + "',0,'"+ DateTime.Now.ToString() +"')";
                DB db = new DB();
                db.ExecuteSQLNonQuery(sqlStr);
            }
            public void modify()//个人信息维护
            {
                List<SqlParameter> sqlParams = new List<SqlParameter>();
                string sqlStr = "UPDATE    Z_USER SET user_name =
@name,user_sex=@sex ,user_email=@email "
                    +
",user_address=@address ,user_info=@info,user_telephone=@telephone ,user_birthday=@birthday "
                    + ",user_photo=@user_photo WHERE user_id=" + user_id.ToString();
                sqlParams.Add(new SqlParameter("@name", user_name));
                sqlParams.Add(new SqlParameter("@sex", user_sex));
                sqlParams.Add(new SqlParameter("@email", user_email));
                sqlParams.Add(new SqlParameter("@address", user_address));
                sqlParams.Add(new SqlParameter("@info", user_info));
                sqlParams.Add(new SqlParameter("@telephone", user_telephone));
                if (user_birthday.Year ==1)
                    sqlParams.Add(new SqlParameter("@birthday", DBNull.Value));
                else
                    sqlParams.Add(new SqlParameter("@birthday", user_birthday));
                if(user_photo==null)
                    sqlParams.Add(new SqlParameter("@user_photo",DBNull.Value));
                else
                    sqlParams.Add(new SqlParameter("@user_photo",user_photo));
                DB db = new DB();
                db.ExecuteSQLNonQueryWithParam(sqlStr,sqlParams);
            }
```

2. 消息实体类 Zmessage.cs 类

在 App_Code 目录中添加新的类文件 Zmessage.cs。在该类中添加操作 Z_MESSAGE 表对应的自动属性已经错误反馈属性 errorMsg。相应的代码如下：

```
            public int msg_id { get; set; }
```

```
            public int user_id { get; set; }
            public string msg_content { get; set; }
            public int reply_count { get; set; }
            public DateTime post_time { get; set; }
            public string errorMsg { get; set; } //用于存放错误消息
```

与消息相关的操作主要有如下几个。

- 获取消息，包括获取所有用户的消息、获取指定用户的消息和获取登录用户及其收听用户的消息。
- 删除自己的消息。
- 转播或评论消息。
- 获取某消息的所有转播或评论记录。

相应方法的实现如下所示：

```
            public DataTable getTopMsg()//获取最新消息，供未登录用户查看
            {
                string sqlStr = "SELECT TOP 20 [msg_id],[msg_content], [user_name],
[user_login],[post_time], [reply_count] FROM [Z_MESSAGE],[Z_USER] WHERE
Z_MESSAGE.user_id = Z_USER.user_id ORDER BY msg_id DESC";
                DB db = new DB();
                return db.GetDataTable(sqlStr);
            }
            public DataTable getUserAndOtherMsg(string user_id)//登录用户以及其收听的用户发表的消息
            {
                string sqlStr = "SELECT TOP 20 [msg_id],[msg_content], [user_name],
[user_login],[post_time], [reply_count] FROM [Z_MESSAGE],[Z_USER] WHERE
Z_MESSAGE.user_id = Z_USER.user_id AND (Z_MESSAGE.user_id=" + user_id + " OR
Z_MESSAGE.user_id IN (SELECT fun_user_id FROM Z_USER_FUN WHERE user_id =" + user_id +
"))ORDER BY msg_id DESC";
                DB db = new DB();
                return db.GetDataTable(sqlStr);
            }
            public DataTable getUserMsg(string user_id)//指定用户的消息
            {
                string sqlStr = "SELECT TOP 20 [msg_id],[msg_content], [user_name],[user_login],
[post_time], [reply_count] FROM [Z_MESSAGE],[Z_USER] WHERE Z_MESSAGE.user_id =
Z_USER.user_id AND Z_MESSAGE.user_id=" + user_id + " ORDER BY msg_id DESC";
                DB db = new DB();
                return db.GetDataTable(sqlStr);
            }
            public bool delMsg(string msgId, string userId)//删除自己发表的消息
            {
                bool ret = true;
                string sqlStr = "SELECT * FROM [Z_MESSAGE] WHERE user_id = " + userId + " AND
```

```
msg_id =" + msgId;
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            if (dr == null)
            {
                errorMsg = "您不能删除别人的消息";
                return false;
            }
            //先删除 Z_REPLY 表中的对应评论，然后删除 Z_MESSAGE 表
            db.BeginTrans();
            try
            {
                sqlStr = "DELETE  FROM [Z_REPLY] WHERE msg_id =" + msgId;
                db.ExecuteSQLNonQuery(sqlStr);
                sqlStr = "DELETE  FROM [Z_MESSAGE] WHERE msg_id =" + msgId;
                db.ExecuteSQLNonQuery(sqlStr);
            }
            catch (Exception e)
            {
                errorMsg = e.Message;
                db.Rollback();
                ret = false;
            }
            finally
            {
                db.close();
            }
            return ret;
        }
        public void replyMsg(string uid,string msgId,string strMsg)//转播或评论消息
        {
            string sqlStr = "SELECT user_id FROM Z_MESSAGE WHERE msg_id =" + msgId ;
            DB db = new DB();
            DataRow dr = db.GetDataRow(sqlStr);
            string srcUserId = dr["user_id"].ToString();
            db.open();
            db.BeginTrans();
            try
            {//先插入 Z_REPLY 表，然后更新 Z_MESSAGE 表中的 reply_count 字段
                sqlStr = "INSERT INTO
Z_REPLY(msg_id,reply_user_id,src_user_id,reply_content,reply_time) VALUES(" +
                    msgId + "," + uid + "," + srcUserId + ",'" + strMsg + "','" +
DateTime.Now.ToString() + "')";
                db.ExecuteSQLNonQueryWithTran(sqlStr);
```

```
                sqlStr = "UPDATE Z_MESSAGE SET reply_count = reply_count+1 WHERE
msg_id=" + msgId;
                db.ExecuteSQLNonQueryWithTran(sqlStr);
                db.Commit();
            }
            catch (Exception e)
            {
                db.Rollback();
            }
            finally
            {
                db.close();
            }
        }
        public DataTable getReply(string msgId)//获取消息的所有转播和评论记录
        {
            string sqlStr = "SELECT [reply_content], [user_name],[reply_time] FROM
Z_REPLY,Z_USER WHERE msg_id =" +
                msgId + " AND   Z_REPLY.reply_user_id = Z_USER.user_id UNION "+
                " SELECT msg_content AS reply_content, user_name,post_time AS reply_time
FROM Z_MESSAGE,Z_USER " +
                " WHERE msg_id=" + msgId + " AND Z_MESSAGE.user_id = Z_USER.user_id
ORDER BY reply_time DESC";
            DB db = new DB();
            return db.GetDataTable(sqlStr);
        }
```

12.2.3　设计母版页

为了使网站的所有页面都具有相同的布局风格和外观，需要添加母版页，然后基于该母版页创建其他页面。

新建母版页，名称为 MasterPage.master。在母版页中有两个 ContentPlaceHolder 控件，一个位于<head>标记中，一个位于<form>中。删除<head>标记中的<title>标记，这样，在内容页中可以设置每个内容页的标题信息。

1. 页面设计

首先创建一个样式表文件，并在母版页中引入该样式表文件，所有的内容页都可以应用该样式表文件中的样式定义。

本例中新建的样式表文件名为 StyleSheet.css，内容如下：

```
body
{
 width: 800px;
```

```
 background-color: #f3f3f3;
}
img
{
 overflow: auto;
}
h1
{
 color: #808000;
}
#left, #right
{
 background-color: #eeeeee;
 border: 1px solid #C0C0C0;
 height: 500px;
}
#left
{
 width: 240px;
 float: left;
 height: 400px;
}
```

在母版页的<head>标记中添加如下代码引入该样式表文件：

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

整个网站的布局设置为头部、中间内容区域和页尾部分。母版页中需要设计的是头部和尾部，中部保留 ContentPlaceHolder 控件不变即可。首先，添加一个<table>通过 1 个图片来显示网站的标题图片，代码如下：

```
<table width=800 border="0" cellspacing="0" cellpadding="0" >
    <tr>
        <td width="100%"><img src="images/bg.jpg" width="100%" height="180"
/></td>
    </tr>
</table>
```

接下来，添加两个 Panel 控件，分别用于显示未登录用户的快速登录和登录用户的欢迎信息。这两个 Panel 控件在同一时刻将只有一个可见，代码如下：

```
<asp:Panel ID="Panel1" runat="server" Height="50px" Width="100%">
    <table width="100%" style="background-color:#f0f0f0;" border="0" cellpadding="0"
cellspacing="0">
    <tr>
        <td style="height: 37px; font-weight: bolder; color: #800080;" align="left">
            <div><a href="Index.aspx">首页</a></div>
```

```
                    </td>
                    <td align="right" width="80%" style="height: 25px">
                        <div>
                            <asp:Label ID="lblInfo" runat="server" Text="用户名"
Width="122px"></asp:Label>
                            <asp:TextBox ID="userLogin" runat="server" Width="80px"
></asp:TextBox> 
                            <asp:Label ID="Label1" runat="server" Text="密码"
Width="42px"></asp:Label>
                            <asp:TextBox ID="userPwd" runat="server" Width="80px"
TextMode="Password"></asp:TextBox> 
                            <asp:Button ID="btnLogin" runat="server" Text="登录" Width="80px"
OnClick="btnLogin_Click" /> 
                            <a href="Register.aspx">注册</a></div>
                    </td>
                </tr>
                </table>
            </asp:Panel>
            <asp:Panel ID="Panel2" runat="server" Height="50px" Width="100%">
                <table width="100%" style="background-color:#f5f5f5;" border="0" cellpadding="0"
cellspacing="0">
                <tr>
                    <td style="height: 37px;    font-weight: bolder; color: #800080;" align="left">
                        <div><a href="Index.aspx">我的首页</a></div>
                    </td>
                    <td align="right"><%= Session["user_name"] %> 欢迎您的光临 <a
href="Search.aspx">找人</a></td>
                </tr>
                </table>
            </asp:Panel>
```

在</form>结束标记的上方，添加页尾部分，代码如下：

```
        <div>
            <hr />
            <table width="100%" border="0" cellspacing="0" cellpadding="0"
style="background-color:#f0f0f0;">
                <tr>
                    <td align="center" width="0%">版权所有(C) 金百合迷你微博 2012</td>
                </tr>
            </table>
        </div>
```

注意：

ContentPlaceHolder1 中不要添加任何代码，这是为内容页预留的占位符。

2. 后台代码

在母版页的后台代码中需要实现如下功能：加载页面时，根据用户当前是否登录显示或隐藏相应的 Panel 控件，实现用户的快速登录功能；登录成功后，将用户信息保存至 Session 变量中。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["user_login"] == null)//如果用户没有登录
    {//显示 Panel1，隐藏 Panel2
        Panel1.Visible = true;
        Panel2.Visible = false;
    }
    else
    {//登录成功隐藏 Panel1，显示 Panel2
        Panel1.Visible = false;
        Panel2.Visible = true;
    }
}
protected void btnLogin_Click(object sender, EventArgs e)
{
    Zuser user = new Zuser();
    if (user.login(userLogin.Text, userPwd.Text))
    {//设置 Session;
        Session["user_login"] = userLogin.Text;
        Session["user_id"] = user.user_id;
        Session["user_name"] = user.user_name;
        Session["user"] = user;
        Panel2.Visible = true;
        Panel1.Visible = false;
        Response.Redirect(Request.Path + "?" + Request.QueryString);
    }
    else
    {
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning",
"alert(\""+user.errorMsg+"\");", true);
    }
}
```

至此，完成母版页的设计，后面创建的所有页面都是基于此母版页的。

12.2.4　首页

网站的首页为 Index.aspx，在该页面中将显示最新的用户消息以及登录用户的相关信息。

1. 页面总体设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
<title>我的首页</title>
</asp:Content>
```

由于该页需要多次后台交互，所以设计为异步刷新，在内容区域中首先添加 ScriptManager 和 UpdatePanel 控件。后面添加的控件都放置在 UpdatePanel 控件内。

该页面通过两个<div>来进行布局，一个 id 为 left，一个 id 为 right，左边的<div>包含一个 Panel 控件，用于显示登录用户的相关信息；右边的<div>包含一个 MultiView 控件，该 MultiView 控件中包含两个 View 控件，其中 View1 包含两个 Panel 控件，一个用于发表新消息，一个用于显示已发表的消息，View2 用于显示用户的听众或用户收听的用户列表。在“源”视图中，可以看到页面的代码层次结构如图 12-2 所示。

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Ru
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
    <div id="left">
        <asp:Panel ID="Panel1">...</asp:Panel>
    </div>
    <div id="right">
        <asp:MultiView ID="MultiView1" runat="server">
            <asp:View ID="View1" runat="server">
                <asp:Panel ID="Panel2">...</asp:Panel>
                <asp:Panel ID="Panel3">...</asp:Panel>
            </asp:View>
            <asp:View ID="View2" runat="server">
                <asp:Panel ID="Panel4">...</asp:Panel>
            </asp:View>
        </asp:MultiView>
    </div>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

图 12-2　首页的代码层次结构

2. 左侧<div>设计

id 为 left 的左侧<div>包含 Panel1 控件，该控件中的设计和布局类似于第 5 章中的例 5-7 中的 Panel2，相应的源代码如下：

```
<asp:Panel ID="Panel1" runat="server">
    <table class="style1" border="1" cellpadding="1">
        <tr align="center">
            <td colspan="3">
                <asp:Image ID="Image1" runat="server" ImageUrl="~/images/boy.gif" />
                <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/Modify.aspx">维护个人信息</asp:HyperLink>
            </td>
        </tr>
```

```
                    <tr align="center">
                        <td>收听</td>
                        <td>听众</td>
                        <td>信息</td>
                    </tr>
                    <tr align="center">
                        <td>
                            <asp:LinkButton ID="LinkButtonOther" runat="server" Text="Other"
                                onclick="LinkButtonOther_Click"></asp:LinkButton>
                        </td>
                        <td>
                            <asp:LinkButton ID="LinkButtonFun" runat="server" Text="Fun"
                                onclick="LinkButtonFun_Click"></asp:LinkButton>
                        </td>
                        <td>
                            <asp:LinkButton ID="LinkButtonMsg" runat="server" Text="Msg"
                                onclick="LinkButtonMsg_Click"></asp:LinkButton>
                        </td>
                    </tr>
                    <tr>
                        <td align="center">姓名</td>
                        <td colspan="2">
                            <asp:Label ID="LabelName" runat="server" Text="" ></asp:Label>
                        </td>
                    </tr>
                    <tr>
                        <td align="center">Email</td>
                        <td colspan="2">
                            <asp:Label ID="LabelEmail" runat="server" Text=" "></asp:Label>
                        </td>
                    </tr>
                    <tr>
                        <td align="center">所在地</td>
                        <td colspan="2">
                            <asp:Label ID="LabelAddress" runat="server" Text=" "></asp:Label>
                        </td>
                    </tr>
                    <tr>
                        <td align="center">电话</td>
                        <td colspan="2">
                            <asp:Label ID="LabelTelephone" runat="server" Text="
"></asp:Label>
                        </td>
                    </tr>
```

```
                <tr>
                    <td align="center">个人简介</td>
                    <td colspan="2">
                        <asp:Label ID="LabelInfo" runat="server"
Text="LabelInfo"></asp:Label>
                    </td>
                </tr>
            </table>
        </asp:Panel>
```

3. 右侧<div>中的 View1 设计

View1 中包含两个 Panel 控件：Panel2 和 Panel3。其中，Panel2 包含一个文本框、一个按钮和一个提示信息用的 Label 控件，相应的代码如下：

```
<asp:Panel ID="Panel2" runat="server">
    <asp:TextBox ID="TextBoxMsg" runat="server" Height="65px" TextMode="MultiLine"
        Width="453px"></asp:TextBox>
    <asp:Button ID="ButtonPost" runat="server"
        Text="发表" onclick="ButtonPost_Click" style="height: 21px" /><br />
    <asp:Label ID="LabelMsg" runat="server" Text="Label"></asp:Label>
</asp:Panel>
```

Panel3 中包含一个 ListView 控件，该控件用于显示 Z_MESSAGE 表中的消息，根据场景不同，可能显示的内容也不同(分别对应 Zmessage 类中的 getTopMsg、getUserAndOtherMsg 和 getUserMsg 方法返回的数据)。这里的 ListView 控件选择为“项目符合列表”布局方式，数据源需要通过后台编码指定，在“源”视图中需要手动编辑控件的< AlternatingItem-Template>和< ItemTemplate>模板，通过 Eval 方式绑定数据集中的字段，相应的代码如下：

```
<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1" >
    <AlternatingItemTemplate>
        <li style="background-color: #FFF8DC;">
            <asp:Label ID="userameLabel" runat="server" Text='<%# Eval("user_name") %>' />:
            <asp:Label ID="msg_contentLabel" runat="server"
                Text='<%# Eval("msg_content") %>' />
            <br />          发表时间:
            <asp:Label ID="post_timeLabel" runat="server" Text='<%# Eval("post_time") %>' />
            <br />           评论次数:
            <asp:Label ID="reply_countLabel" runat="server"
                Text='<%# Eval("reply_count") %>' />
            <asp:LinkButton ID="replyHyper" runat="server" CommandArgument='<%#
Eval("msg_id") %>' Visible='<%# Session["user_id"]!=null %>'   OnClick="replyMsg" Text="转播" />
            <asp:LinkButton ID="delLinkButton" runat="server" CommandArgument='<%#
Eval("msg_id") %>' Visible='<%# Session["user_login"]!=null && Session["user_login"].ToString() ==
Eval("user_login").ToString() %>'   OnClick="delMsg" Text="删除" />
```

```
                <br /><br />
            </li>
        </AlternatingItemTemplate>
        <EmptyDataTemplate>
            没有任何消息。
        </EmptyDataTemplate>
        <ItemTemplate>
            <li style="background-color: #DCDCDC;color: #000000;">
                <asp:Label ID="uameLabel" runat="server" Text='<%# Eval("user_name") %>' />:
                <asp:Label ID="msg_contentLabel" runat="server"
                    Text='<%# Eval("msg_content") %>' />
                <br />          发表时间:
                <asp:Label ID="post_timeLabel" runat="server" Text='<%# Eval("post_time") %>' />
                <br />          评论次数:
                <asp:Label ID="reply_countLabel" runat="server"
                    Text='<%# Eval("reply_count") %>' />
                <asp:LinkButton ID="replyHyper" runat="server" CommandArgument='<%#
Eval("msg_id") %>' Visible='<%# Session["user_id"]!=null %>' OnClick="replyMsg" Text="转播" />
                <asp:LinkButton ID="delLinkButton" runat="server" CommandArgument='<%#
Eval("msg_id") %>' Visible='<%# Session["user_login"]!=null && Session["user_login"].ToString() ==
Eval("user_login").ToString() %>'   OnClick="delMsg" Text="删除" />
                <br /><br />
            </li>
        </ItemTemplate>
        <LayoutTemplate>
            <ul ID="itemPlaceholderContainer" runat="server"
                style="font-family: Verdana, Arial, Helvetica, sans-serif;">
                <li runat="server" id="itemPlaceholder" />
            </ul>
            <div style="text-align: center;background-color: #CCCCCC;font-family: Verdana,
Arial, Helvetica, sans-serif;color: #000000;">
            </div>
        </LayoutTemplate>
    </asp:ListView>
```

4. 右侧<div>中的 View2 设计

View2 中包含一个 ListView 控件，该控件用于显示用户收听的用户列表或者用户的听众列表(分别对应 Zuser 类中的 getUserOther 和 getUserFun 方法返回的数据)，这个 ListView 控件选择为“平铺”布局方式，数据源需要通过后台编码指定，在“源”视图中需要手动编辑控件的< AlternatingItemTemplate>和< ItemTemplate>模板，通过 Eval 方式绑定数据集中的字段，相应的代码如下：

```
        <asp:ListView ID="ListView2" runat="server" DataSourceID="SqlDataSource2"
            DataKeyNames="user_id" GroupItemCount="3">
            <AlternatingItemTemplate>
                <td runat="server" style="background-color: #FFFFFF;color: #284775;">
                    <asp:Image ID="Image2" runat="server" ImageUrl='<%# Eval("photo_path") %>' />
                    <br />
                    <asp:LinkButton ID="Button1" runat="server" Text='<%# Eval("user_name") %>'
                        OnClick="ShowUser" CommandArgument='<%# Eval("user_id") %>' />(
                    <asp:Label ID="user_login" runat="server" Text='<%# Eval("user_login") %>' />)
                    <br />         所在地:
                    <asp:Label ID="user_addressLabel" runat="server"
                        Text='<%# Eval("user_address") %>' />
                    <br />         生日:
                    <asp:Label ID="birthday" runat="server" Text='<%# Eval("user_birthday") %>' />
                </td>
            </AlternatingItemTemplate>
            <EmptyDataTemplate>
                <table runat="server"
                    style="background-color: #FFFFFF;border-collapse: collapse;border-color:
#999999;border-style:none;border-width:1px;">
                    <tr><td>未返回数据。</td>
                    </tr>
                </table>
            </EmptyDataTemplate>
            <GroupTemplate>
                <tr ID="itemPlaceholderContainer" runat="server">
                    <td ID="itemPlaceholder" runat="server">         </td>
                </tr>
            </GroupTemplate>
            <ItemTemplate>
                <td runat="server" style="background-color: #E0FFFF;color: #333333;">
                    <asp:Image ID="Image2" runat="server" ImageUrl='<%# Eval("photo_path") %>' />
                    <br />
                    <asp:LinkButton ID="Button1" runat="server" Text='<%# Eval("user_name") %>'
                        OnClick="ShowUser" CommandArgument='<%# Eval("user_id") %>' />(
                    <asp:Label ID="Label2" runat="server"   Text='<%# Eval("user_login") %>' />)
                    <br />        所在地:
                    <asp:Label ID="Label4" runat="server" Text='<%# Eval("user_address") %>' />
                    <br />         生日:
                    <asp:Label ID="Label5" runat="server"   Text='<%# Eval("user_birthday") %>' />
                </td>
            </ItemTemplate>
            <LayoutTemplate>
                <table runat="server">
```

```
                    <tr runat="server">
                        <td runat="server">
                            <table ID="groupPlaceholderContainer" runat="server" border="1"
                                style="background-color: #FFFFFF;border-collapse:
collapse;border-color: #999999;border-style:none;border-width:1px;font-family: Verdana, Arial, Helvetica,
sans-serif;">
                                <tr ID="groupPlaceholder" runat="server">
                                </tr>
                            </table>
                        </td>
                    </tr>
                    <tr runat="server">
                        <td runat="server"
                            style="text-align: center;background-color: #5D7B9D;font-family:
Verdana, Arial, Helvetica, sans-serif;color: #FFFFFF">
                        </td>
                    </tr>
                </table>
            </LayoutTemplate>
        </asp:ListView>
```

说明：

Image 控件的 ImageUrl 属性绑定的数据集字段是 photo_path，该字段是通过后台的 SELECT 语句用 AS 方法重命名的输出列，详见 Zuser 类的 getUserOther 和 getUserFun 方法。

5. 后台代码

当页面加载时，需要根据用户是否登录来隐藏需要的 Panel 控件，Page_Load 方法的代码如下：

```
        protected void Page_Load(object sender, EventArgs e)
        {
            LabelMsg.Visible = false;
            if (!Page.IsPostBack)
            {
                MultiView1.ActiveViewIndex = 0;
                if (Session["user_login"] == null)//如果用户没有登录
                {//隐藏 Panel1，显示最新消息
                    Panel1.Visible = false;
                    Panel2.Visible = false;
                }
                else
                    showPanel1(Session["user_login"].ToString());
                showMsg();
            }
```

```
        }
        private void showMsg()//显示消息
        {
            Zmessage msg = new Zmessage();
            ListView1.DataSourceID = "";
            if (Session["user_login"] == null)//如果是匿名用户，显示最新消息
            {
                ListView1.DataSource = msg.getTopMsg();
                ListView1.DataBind();
            }
            else
            {
                Panel2.Visible = true;
                ListView1.DataSource = msg.getUserAndOtherMsg(Session["user_id"].ToString());
                ListView1.DataBind();
            }
        }
        private void showPanel1(string userLogin)//Panel1 中加载登录用户信息
        {
            Panel1.Visible = true;
            Zuser user = (Zuser)Session["user"];
            LabelName.Text = user.user_name;
            LabelEmail.Text = user.user_email;
            LabelAddress.Text = user.user_address;
            LabelInfo.Text = user.user_info;
            LabelTelephone.Text =user.user_telephone;
            LinkButtonOther.Text = user.getOther(user.user_id);
            LinkButtonFun.Text = user.getFun(user.user_id);
            LinkButtonMsg.Text = user.getMsg(user.user_id);
            Image1.ImageUrl = "~/touxiang.aspx?userid=" + user.user_id.ToString();
        }
```

说明：

头像的显示与第 5 章所讲的方法类似，也是借助 touxiang.aspx 页面进行加载并显示。本节将会介绍该页面的具体实现。

对于登录成功的用户，个人信息中的 3 个 LinkButton 分别对应用户收听、听众和消息。需要的 Click 事件处理程序如下：

```
        protected void LinkButtonOther_Click(object sender, EventArgs e)
        {
            MultiView1.ActiveViewIndex = 1;//View2 可见
            ListView2.DataSourceID = "";
            Zuser user = new Zuser();
            ListView2.DataSource = user.getUserOther(Session["user_id"].ToString());
```

```
        ListView2.DataBind();
    }
    protected void LinkButtonFun_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 1;//View2 可见
        ListView2.DataSourceID = "";
        Zuser user = new Zuser();
        ListView2.DataSource = user.getUserFun(Session["user_id"].ToString());
        ListView2.DataBind();
    }
    protected void LinkButtonMsg_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 0;//View1 可见
        Zmessage msg = new Zmessage();
        ListView1.DataSource = msg.getUserMsg(Session["user_id"].ToString());
        ListView1.DataBind();
        Panel2.Visible = true;
    }
```

对于登录用户，还可以发表新消息、转播已有消息和删除自己的某条消息，需要的代码如下：

```
    protected void ButtonPost_Click(object sender, EventArgs e)
    {
        if (TextBoxMsg.Text.Length > 140)
        {
            LabelMsg.Text="消息长度最大为 140";
            LabelMsg.Visible = true;
            return;
        }
        if (TextBoxMsg.Text.Length == 0)
        {
            LabelMsg.Text = "请输入消息内容";
            LabelMsg.Visible = true;
            return;
        }
        Zuser user = (Zuser)Session["user"];
        user.postMsg(TextBoxMsg.Text);
        TextBoxMsg.Text = "";
        showMsg();
    }
    protected void replyMsg(object sender, EventArgs e)
    {
        LinkButton lb = (LinkButton)sender;
        string msgId = lb.CommandArgument;
```

```
        string strUrl = "Reply.aspx?msgId="+msgId+"&userId="+Session["user_id"].ToString();
        Response.Redirect(strUrl);
    }
    protected void delMsg(object sender, EventArgs e)
    {
        LinkButton lb = (LinkButton)sender;
        string msgId = lb.CommandArgument;
        Zmessage msg = new Zmessage();
        msg.delMsg(msgId,Session["user_id"].ToString());
        showMsg();
    }
```

技巧：

上述代码中通过 LinkButton 的 CommandArgument 属性传递了参数，该参数在 ListView 控件中是通过 Eval 方式绑定的数据集中的字段 msg_id。

当右侧 View2 控件可见时，ListView2 控件显示了用户收听或听众信息，单击某个用户，可跳转到相应的“个人资料”页面，代码如下：

```
    protected void ShowUser(object sender, EventArgs e)
    {
        LinkButton lb = (LinkButton)sender;
        string userId = lb.CommandArgument;
        string strUrl = "UserInfo.aspx?userId=" + userId;
        Response.Redirect(strUrl);
    }
```

6. 显示头像辅助页 touxiang.aspx

通过“添加新项”对话框添加 touxiang.aspx 页面，需要注意的是，这个页面不要使用母版页。在页面的 Load 事件中，根据请求中的 QueryString 集合获取用户 ID，然后判断是否有专属头像，如果没有则根据性别显示系统默认头像，Page_Load 方法的代码如下：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        Zuser user = new Zuser();
        byte[] data = user.getPhotoById(Request.QueryString["userid"].ToString());
        if (data == null)
        {
            string fileName = Request.PhysicalApplicationPath + "/images/girl.gif";
            if (user.user_sex == "男")
                fileName = Request.PhysicalApplicationPath + "/images/boy.gif";
            FileStream fs = new FileStream(fileName,FileMode.Open);
            data = new byte[fs.Length+1];
            fs.Read(data, 0, (int)fs.Length);
        }
```

```
            Response.ContentType = "image/gif";
            Response.OutputStream.Write(data, 0, data.Length);
        }
```

至此，完成首页面的全部功能，等全部页面创建完成后，将统一测试页面的运行效果。

12.2.5　注册页面

注册页面 Register.aspx 比较简单，只需提供用户注册所需的表单即可。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content2" ContentPlaceHolderID="head" Runat="Server">
    <title>注册微博新用户</title>
</asp:Content>
```

在内容区域添加注册表单，与第 5 章中的例 5-6 类似，页面布局如图 12-3 所示。

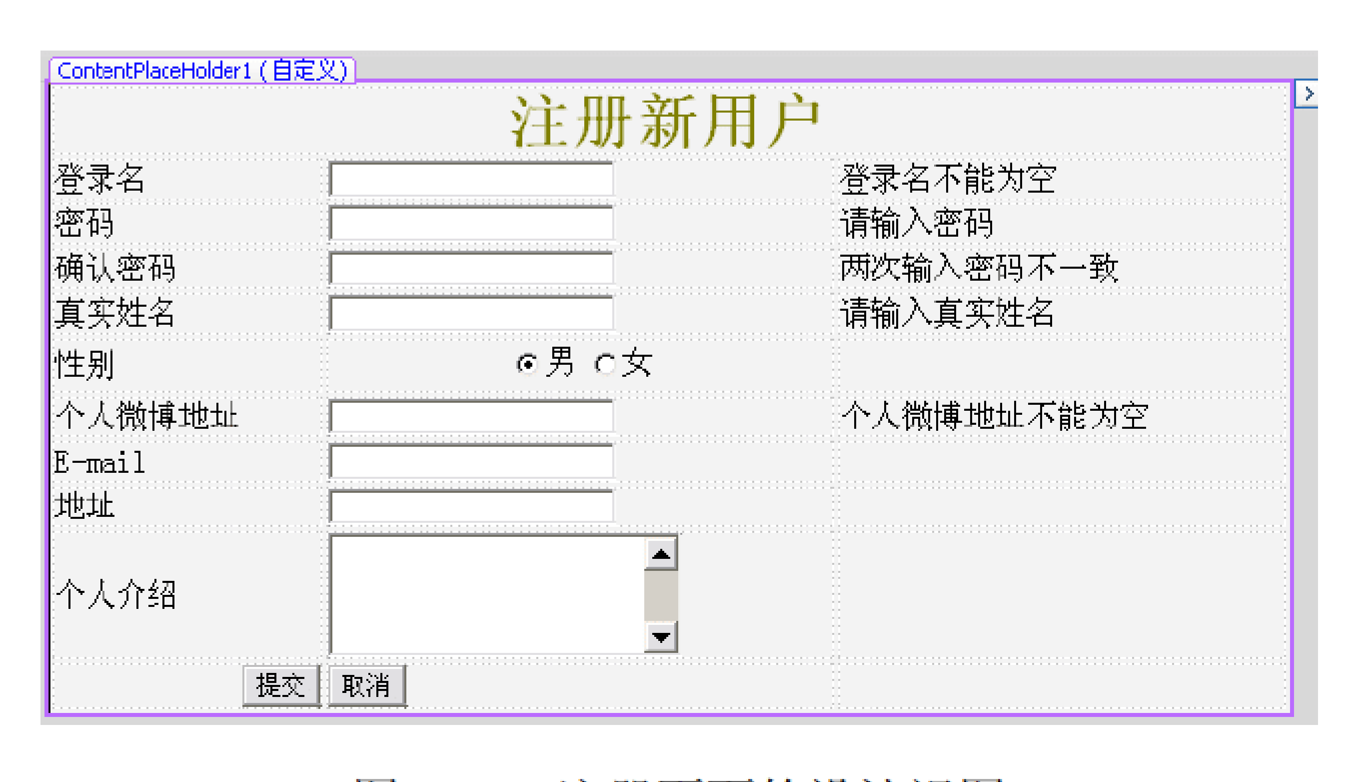


图 12-3　注册页面的设计视图

2. 后台代码

“提交”按钮的单击事件处理程序如下：

```
        protected void ButtonSubmit_Click(object sender, EventArgs e)
        {
            Zuser user = new Zuser();
            if (user.checkLogin(TextBoxLogin.Text.Trim()))
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"该登
录名已存在\"); ", true);
                return;
            }
            if (user.checkUrl(TextBoxHomeUrl.Text))
            {
```

```
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"个人
微博地址已被使用\"); ", true);
            return;
        }
        user.user_name = TextBoxName.Text;
        user.user_login = TextBoxLogin.Text;
        user.user_password = TextBoxPassword.Text;
        user.user_sex = RadioButtonList1.SelectedValue;
        user.home_url = TextBoxHomeUrl.Text;
        user.user_email = TextBoxEmail.Text;
        user.user_info = TextBoxInfo.Text;
        user.user_address = TextBoxAddress.Text;
        user.regist_time = DateTime.Now;
        bool result = user.register();
        if (result)
        {
            user.login(TextBoxLogin.Text,TextBoxPassword.Text);
            Session["user_login"] = TextBoxLogin.Text;
            Session["user_id"] = user.user_id;
            Session["user"] = user;
            Response.Redirect("Index.aspx");
        }
    }
```

12.2.6　查找用户页面

查找用户页面 Search.aspx 用于根据姓名和所在地进行模糊查询，查找自己感兴趣的用户，从而进行收听等操作。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title>我要找人</title>
</asp:Content>
```

内容区域中是一个<table>，表格的前两行是查询条件，第三行是一个 ListView 控件，如图 12-4 所示。ListView 控件用于显示查询结果，该控件的设计与 Index.aspx 页面中的 ListView2 非常相似，这里不再给出源代码。

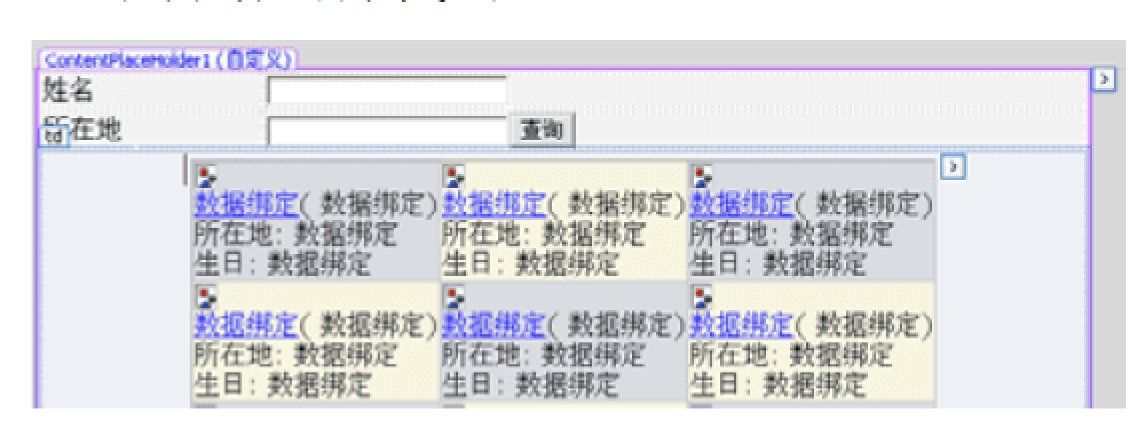


图 12-4　查找用户页面的设计视图

2. 后台代码

该页在加载时，将默认显示所有用户列表，然后判断用户是否登录，如果没有登录，则查询按钮不可用。Page_Load 事件处理函数的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    ListView1.DataSourceID = "";
    Zuser user = new Zuser();
    ListView1.DataSource = user.searchUser(TextBoxName.Text, TextBoxAddress.Text);
    ListView1.DataBind();
    if (Session["user_id"] == null)
        ButtonSearch.Enabled = false;
}
```

“查询”按钮和链接到每个用户信息页的 LinkButton 按钮的 Click 事件处理程序如下：

```
protected void ButtonSearch_Click(object sender, EventArgs e)
{
    if (TextBoxName.Text == "")
    {
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"请输
入姓名\");", true);
        return;
    }
    ListView1.DataSourceID = "";
    Zuser user = new Zuser();
    ListView1.DataSource = user.searchUser(TextBoxName.Text,TextBoxAddress.Text);
    ListView1.DataBind();
}
protected void ShowUser(object sender, EventArgs e)
{
    LinkButton lb = (LinkButton)sender;
    string userId = lb.CommandArgument;
    string strUrl = "UserInfo.aspx?userId=" + userId;
    Response.Redirect(strUrl);
}
```

12.2.7　个人资料页面

个人资料页面 UserInfo.aspx 与 Index.aspx 页面非常相似，只是该页显示的不是当前登录用户的信息，而是查询出来的某个普通用户的基本信息以及他所发表的消息。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，通过 Session 变

量获取要显示的用户姓名，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
<title><%= Session["current_name"].ToString() %> 的个人资料</title>
</asp:Content>
```

内容区域与 Index.aspx 页面非常相似，也是需要 ScriptManager 和 UpdatePanel 控件，然后通过两个<div>来进行布局，所不同的主要有如下两点。

- 左侧<dif>中的 Panel 控件显示的查询的某个用户信息，头像旁边是一个 LinkButton 控件，根据登录用户是否收听了该用户而显示为“收听”或者“取消收听”。
- 右侧 View1 控件中没有用于发表新消息的 Panel 控件，该页中 View1 控件只用于显示指定用户的消息。

这里只给出页面的布局效果图，如图 12-5 所示，具体的控件设置可参考 Index.aspx 页面。

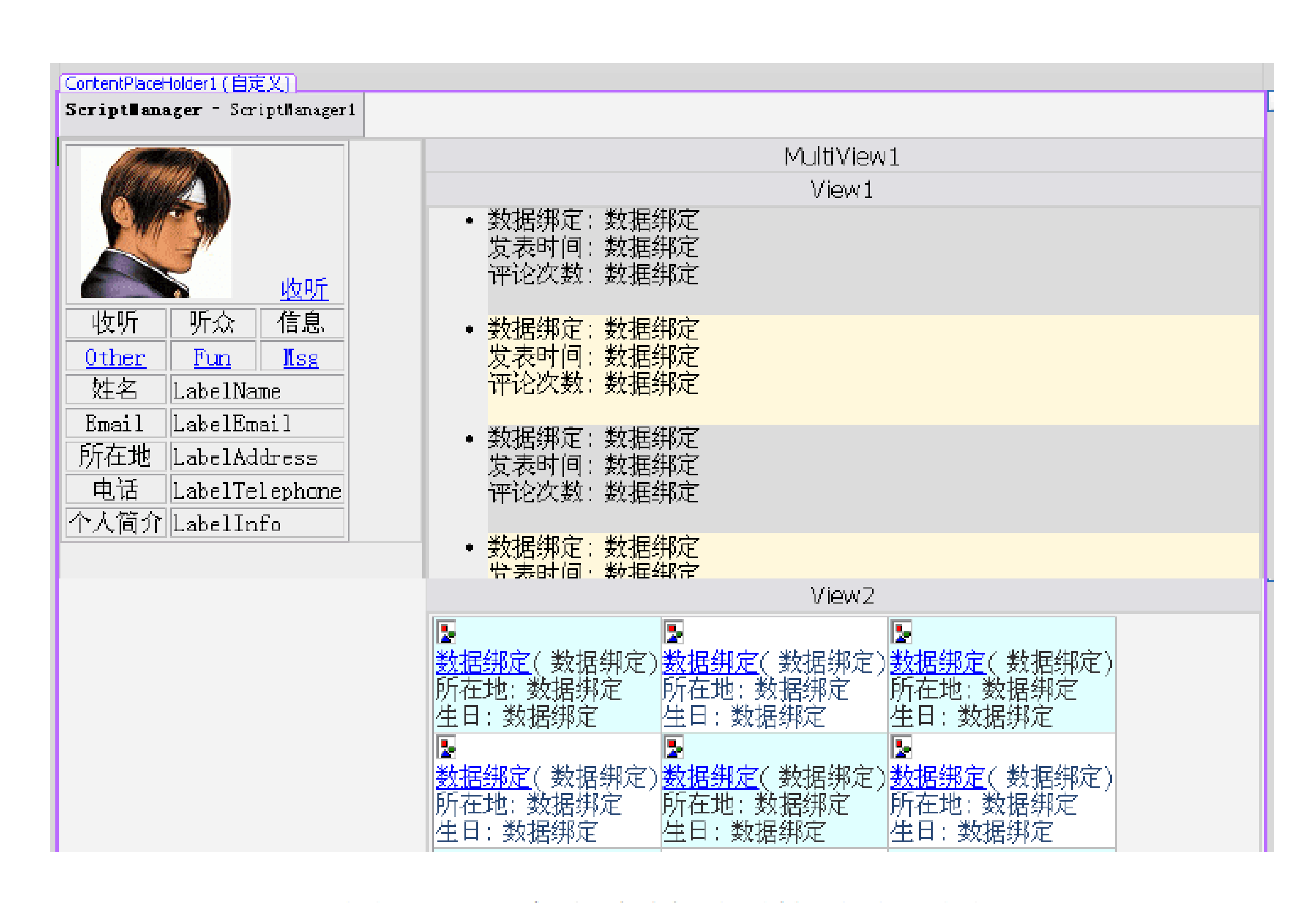


图 12-5 个人资料页面的设计视图

2. 后台代码

在页面的 Load 事件中，通过 QueryString 集合获取要显示用户的 ID，然后将用户的姓名和 ID 存储到 Session 变量中 Session["current_user_id"]和 Session["current_name"]，后面查询收听用户和听众时，将使用这个 Session 变量，完整的后台代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    MultiView1.ActiveViewIndex = 0;
    if (Request.QueryString["userId"] == null)//只显示指定用户信息
    {
        //隐藏 Panel1，显示最新消息
        Panel1.Visible = false;
        Panel2.Visible = false;
    }
    else
```

```
        {
            showPanel1(Request.QueryString["userId"].ToString());
            showMsg(Request.QueryString["userId"].ToString());
            Session["current_user_id"] = Request.QueryString["userId"].ToString();
            Zuser u = new Zuser(Int32.Parse(Request.QueryString["userId"].ToString()));
            Session["current_name"] = u.user_name;
            LinkButton1.Visible=(Session["user_id"]!=null);
            if (Session["user_id"] != null)
            {//判断是否已收听
                Zuser user = new Zuser(Int32.Parse(Session["user_id"].ToString()));
                if (user.hasListen(Request.QueryString["userId"].ToString()))
                    LinkButton1.Text = "取消收听";
                else
                    LinkButton1.Text = "收听";
            }
        }
    }
    private void showPanel1(string userId)
    {
        Panel1.Visible = true;
        Zuser user = new Zuser(Int32.Parse(userId));
        LabelName.Text = user.user_name;
        LabelEmail.Text = user.user_email;
        LabelAddress.Text = user.user_address;
        LabelInfo.Text = user.user_info;
        LabelTelephone.Text = user.user_telephone;
        LinkButtonOther.Text = user.getOther(user.user_id);
        LinkButtonFun.Text = user.getFun(user.user_id);
        LinkButtonMsg.Text = user.getMsg(user.user_id);
        Image1.ImageUrl = "~/touxiang.aspx?userid=" + user.user_id.ToString();
    }
    private void showMsg(string userId)
    {
        Zmessage msg = new Zmessage();
        ListView1.DataSourceID = "";
        Panel2.Visible = true;
        ListView1.DataSource = msg.getUserMsg(userId);
        ListView1.DataBind();
    }
    protected void LinkButtonMsg_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 0;
        Zmessage msg = new Zmessage();
        ListView1.DataSource = msg.getUserMsg(Session["current_user_id"].ToString());
```

```
        ListView1.DataBind();
        Panel2.Visible = true;
    }
    protected void LinkButtonOther_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 1;
        ListView2.DataSourceID = "";
        Zuser user = new Zuser();
        ListView2.DataSource = user.getUserOther(Session["current_user_id"].ToString());
        ListView2.DataBind();
    }
    protected void LinkButtonFun_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 1;
        ListView2.DataSourceID = "";
        Zuser user = new Zuser();
        ListView2.DataSource = user.getUserFun(Session["current_user_id"].ToString());
        ListView2.DataBind();
    }
    protected void ListenUser(object sender, EventArgs e)
    {
        if (LinkButton1.Text == "收听")
        {
            Zuser user = new Zuser(Int32.Parse(Session["user_id"].ToString()));
            user.listen(Session["current_user_id"].ToString());
            LinkButton1.Text = "取消收听";
        }
        else//取消收听
        {
            Zuser user = new Zuser(Int32.Parse(Session["user_id"].ToString()));
            user.cancelListen(Session["current_user_id"].ToString());
            LinkButton1.Text = "收听";
        }
    }
    protected void ShowUser(object sender, EventArgs e)
    {
        LinkButton lb = (LinkButton)sender;
        string userId = lb.CommandArgument;
        string strUrl = "UserInfo.aspx?userId=" + userId;
        Response.Redirect(strUrl);
    }
```

12.2.8　个人信息维护页面

个人信息维护页面 Modify.aspx 与第 5 章的例 5-8 类似，所不同的是这里设置为异步刷新，后台的数据库操作封装到实体类中实现。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，值为主题，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
<title>个人信息维护</title>
</asp:Content>
```

内容区域中首先添加 ScriptManager 和 UpdatePanel 控件，然后是一个<table>表单，布局如图 12-6 所示。



图 12-6　个人信息维护页面的设计视图

2. 后台代码

在页面的 Load 事件中，首先加载用户的信息进行显示，当用户修改了个人信息后，单击“提交”按钮时，后台将进行密码校验，然后调用实体类 Zuser 的方法进行修改操作，完整的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    LabelMsg.Visible = false;
    if (!Page.IsPostBack)
    {
        Calendar1.Visible = false;
        if (Session["user_id"] != null)
            ShowInfo();
        else
        {
```

```
                Response.Redirect("Index.aspx");
            }
        }
    }
    private void ShowInfo()
    {
        Zuser user = (Zuser)Session["user"];
        TextBoxName.Text = user.user_name;
        TextBoxEmail.Text = user.user_email;
        TextBoxAddress.Text = user.user_address;
        TextBoxInfo.Text = user.user_info;
        TextBoxTelephone.Text =user.user_telephone;
        TextBoxBirthday.Text = user.user_birthday.ToString();
        foreach (ListItem item in RadioButtonList1.Items)
            if (item.Value == user.user_sex)
                item.Selected = true;
        Image1.ImageUrl = "~/touxiang.aspx?userid=" + user.user_id.ToString();
    }
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        Calendar1.Visible = true;
    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        TextBoxBirthday.Text = Calendar1.SelectedDate.ToString();
        if (!(TextBoxBirthday.Text == ""))
            Calendar1.Visible = false;
    }
    protected void ButtonModify_Click(object sender, EventArgs e)
    {
        if (Session["user_id"] != null)
        {
            if (TextBoxName.Text == "")
            {
                LabelMsg.Text ="姓名不能为空";
                LabelMsg.Visible = true;
                return;
            }
            Zuser user = (Zuser)Session["user"];
            if (TextBoxPassword.Text == user.user_password)
            {
                user.user_name = TextBoxName.Text;
                user.user_sex = RadioButtonList1.SelectedValue;
                user.user_email = TextBoxEmail.Text;
```

```
                    user.user_address = TextBoxAddress.Text;
                    user.user_info = TextBoxInfo.Text;
                    user.user_telephone = TextBoxTelephone.Text;
                    if (TextBoxBirthday.Text != "")
                        user.user_birthday = DateTime.Parse(TextBoxBirthday.Text);
                    if (FileUpload1.HasFile)
                    {
                        byte[] data = new byte[FileUpload1.FileContent.Length + 1];
                        FileUpload1.FileContent.Read(data, 0,
(int)FileUpload1.FileContent.Length);
                        user.user_photo = data;
                    }
                    user.modify();
                    Session["user"] = user;
                }
                else
                {
                    LabelMsg.Text = "密码错误，不允许修改";
                    LabelMsg.Visible = true;
                }
            }
            else
            {
                LabelMsg.Text ="会话已过期，请重新登录";
                LabelMsg.Visible= true;
                Response.Redirect("Index.aspx");
            }
        }
```

12.2.9　转播和评论消息页面

转播和评论消息页面 Reply.aspx 将根据请求参数显示指定消息和对该消息的所有评论，同时，上方提供文本框，用户可以输入转播评论进行再次转播。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
    <asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title>转播消息</title>
    </asp:Content>
```

内容区域中首先添加 ScriptManager 和 UpdatePanel 控件，然后是一个<table>，第一行放置一个 Panel 控件供用户输入新的转播和评论内容；第二行是一个 ListView 控件，用于显示该消息的所有转播和评论信息。<table>元素的完整代码如下：

```
    <table align="center" width="80%">
        <tr>
            <td>
            <asp:Panel ID="Panel1" runat="server">
                <asp:TextBox ID="TextBoxMsg" runat="server" Height="65px"
TextMode="MultiLine"
                    Width="512px"></asp:TextBox>
                <asp:Button ID="ButtonReply" runat="server"
                    Text="转播" onclick="ButtonReply_Click" style="height: 21px" /><br />
                <asp:Label ID="LabelMsg" runat="server" Text="Label"></asp:Label>
            </asp:Panel>
            </td>
        </tr>
        <tr>
            <td>
            <asp:Panel ID="Panel2" runat="server">
                <asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1">
                    <AlternatingItemTemplate>
                        <li style="background-color: #FFF8DC;">
                            <asp:Label ID="una " runat="server" Text='<%# Eval("user_name") %>' />:
                            <asp:Label ID="msg_contentLabel" runat="server"
                                Text='<%# Eval("reply_content") %>' />
                            <br />                    转播时间:
                            <asp:Label ID="post" runat="server" Text='<%# Eval("reply_time") %>' />
                            <br /><br />
                        </li>
                    </AlternatingItemTemplate>
                    <EmptyDataTemplate>
                        没有任何消息。
                    </EmptyDataTemplate>
                    <ItemTemplate>
                        <li style="background-color: #DCDCDC;color: #000000;">
                            <asp:Label ID="una" runat="server" Text='<%# Eval("user_name") %>' />:
                            <asp:Label ID="msg_contentLabel" runat="server"
                                Text='<%# Eval("reply_content") %>' />
                            <br />                    转播时间:
                            <asp:Label ID="post" runat="server" Text='<%# Eval("reply_time") %>' />
                            <br /><br />
                        </li>
                    </ItemTemplate>
                    <LayoutTemplate>
                        <ul ID="itemPlaceholderContainer" runat="server"
                            style="font-family: Verdana, Arial, Helvetica, sans-serif;">
                            <li runat="server" id="itemPlaceholder" />
```

```
                        </ul>
                        <div style="text-align: center;background-color: #CCCCCC;font-family:
Verdana, Arial, Helvetica, sans-serif;color: #000000;">
                        </div>
                    </LayoutTemplate>
                </asp:ListView>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                    ConnectionString="<%$ ConnectionStrings:WeiBoDb %>"
                    SelectCommand="SELECT [reply_content], [user_name],[reply_time] FROM
[Z_REPLY],[Z_USER] WHERE Z_REPLY.src_user_id = Z_USER.user_id">
                </asp:SqlDataSource>
            </asp:Panel>
            </td>
        </tr>
    </table>
```

2. 后台代码

后台代码主要包括，在页面的 Load 事件中，加载并显示该消息和该消息的所有转播与评论信息；单击“转播”按钮后，调用 Zmessage 类的方法添加新的评论记录，并更新 Z_MESSAGE 表中该消息的 reply_count 字段。完整的代码如下：

```
        protected void Page_Load(object sender, EventArgs e)
        {
            LabelMsg.Visible = false;
            if (Request.QueryString["msgId"] == null)
            {
                Response.Redirect("Index.aspx");
                return;
            }
            Session["msg_id"] = Request.QueryString["msgId"].ToString();
            showReply();
        }
        private void showReply()
        {
            ListView1.DataSourceID = "";
            Zmessage msg = new Zmessage();
            ListView1.DataSource = msg.getReply(Session["msg_id"].ToString());
            ListView1.DataBind();
        }
        protected void ButtonReply_Click(object sender, EventArgs e)
        {
            if (TextBoxMsg.Text.Length > 140)
            {
                LabelMsg.Text = "消息长度最大为 140";
                LabelMsg.Visible = true;
                return;
```

```
            }
            Zmessage msg = new Zmessage();

msg.replyMsg(Session["user_id"].ToString(),Session["msg_id"].ToString(),TextBoxMsg.Text);
            TextBoxMsg.Text = "";
            showReply();
        }
```

至此，已完成所有的页面设计，接下来将测试网站，查看网站运行效果。

12.3　系统运行效果

本节将运行网站，测试网站的各项功能，查询微博系统的运行效果。

12.3.1　设置启动选项

此时的微博系统包含多个页面，接下来，需要设置一下站点的起始页。

选择“网站”|“启动选项”命令，打开站点的属性页，在左侧列表中选择“启动选项”，然后从右侧区域，选择“特定页”单选按钮，单击右侧的“浏览”按钮，打开“选择页码以开始”对话框，这里选择 Index.aspx 页面作为起始页，如图 12-7 所示。

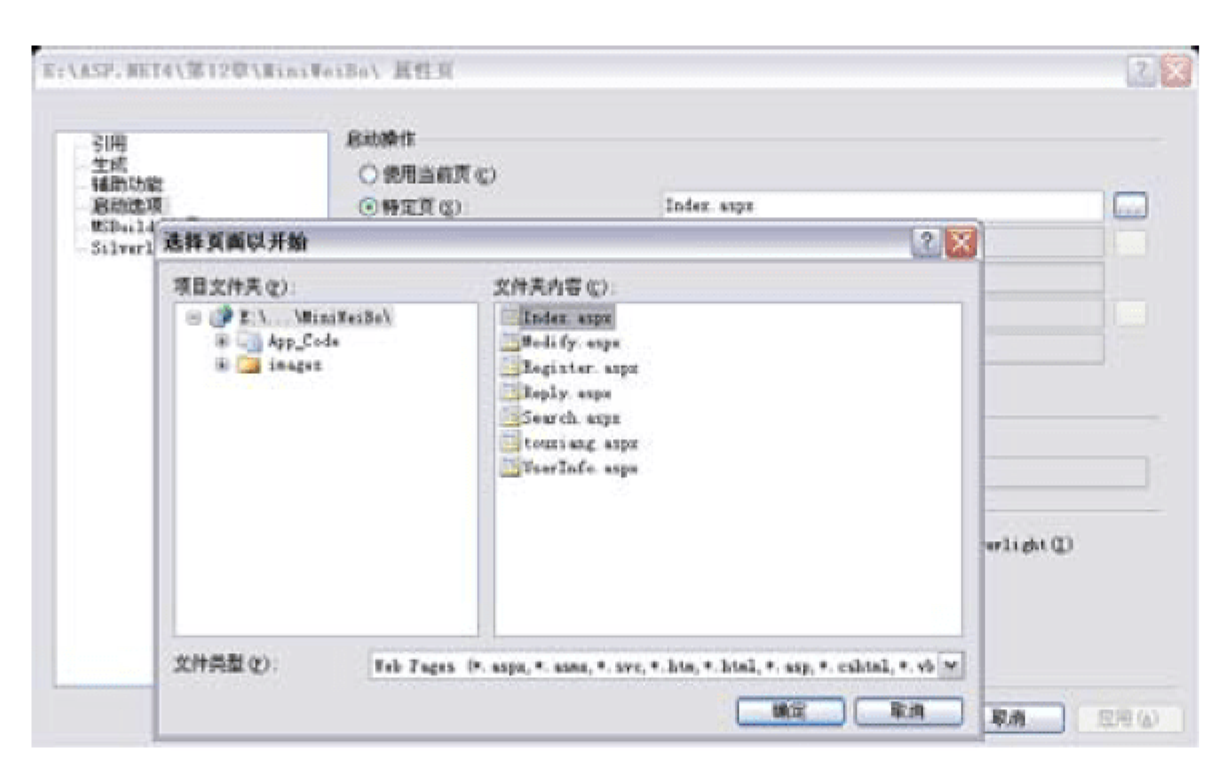


图 12-7　设置站点的起始页

单击“确定”按钮，完成起始页的设置。

12.3.2　测试微博系统的功能

编译并运行程序，在浏览器中打开网站的首页，此时用户尚未登录，将显示系统中最新发表的消息，如图 12-8 所示。

单击“注册”链接可导航到注册页面，进行新用户的注册，如图 12-9 所示。




图 12-8　微博系统首页



图 12-9　注册新用户页面

注册成功后，将进行自动登录并跳转到首页，将显示用户信息和用户默认看到的微博消息，如图 12-10 所示。

此时，可以通过右上角的“找人”链接，查找感兴趣的用户进行收听，如图 12-11 所示。

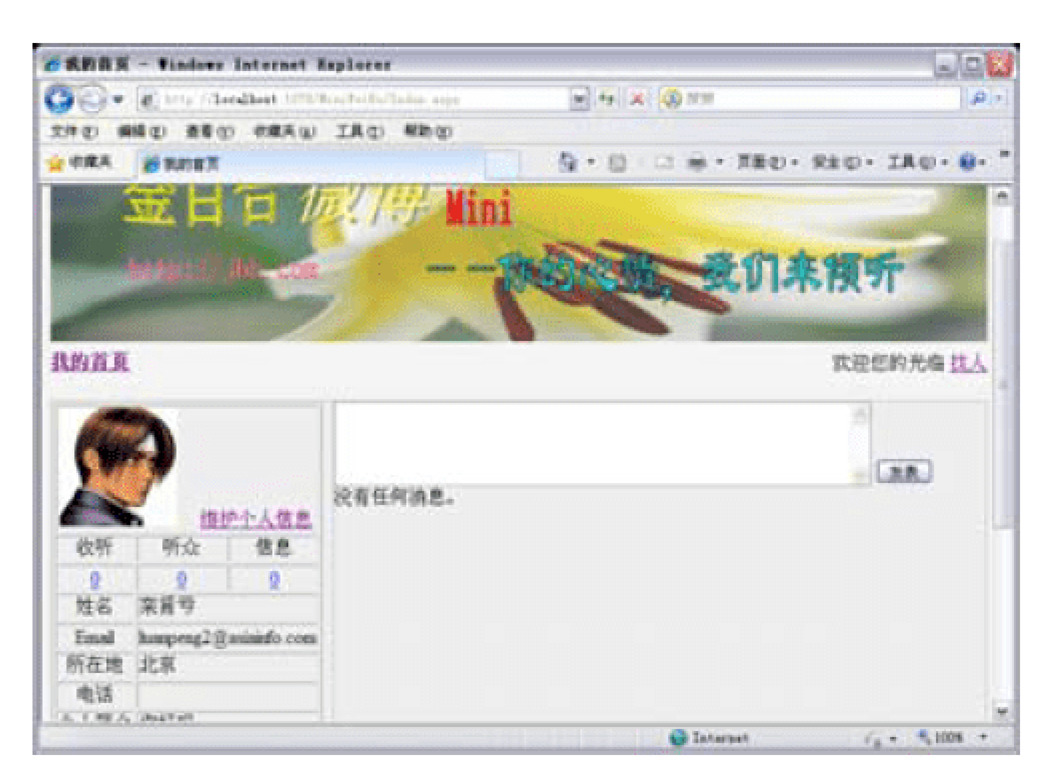


图 12-10　登录后的个人首页



图 12-11　找人页面

单击用户列表中的用户名链接，可进入该用户的个人资料页面，该页显示了该用户收听、听众和发表的消息等信息，如图 12-12 所示。

单击“收听”按钮，可收听该用户。收听后，相应的按钮将变为“取消收听”，单击“我的首页”返回主页面，可以看到收听数发生了变化，右侧窗口也显示了收听用户发表的微博消息，如图 12-13 所示。

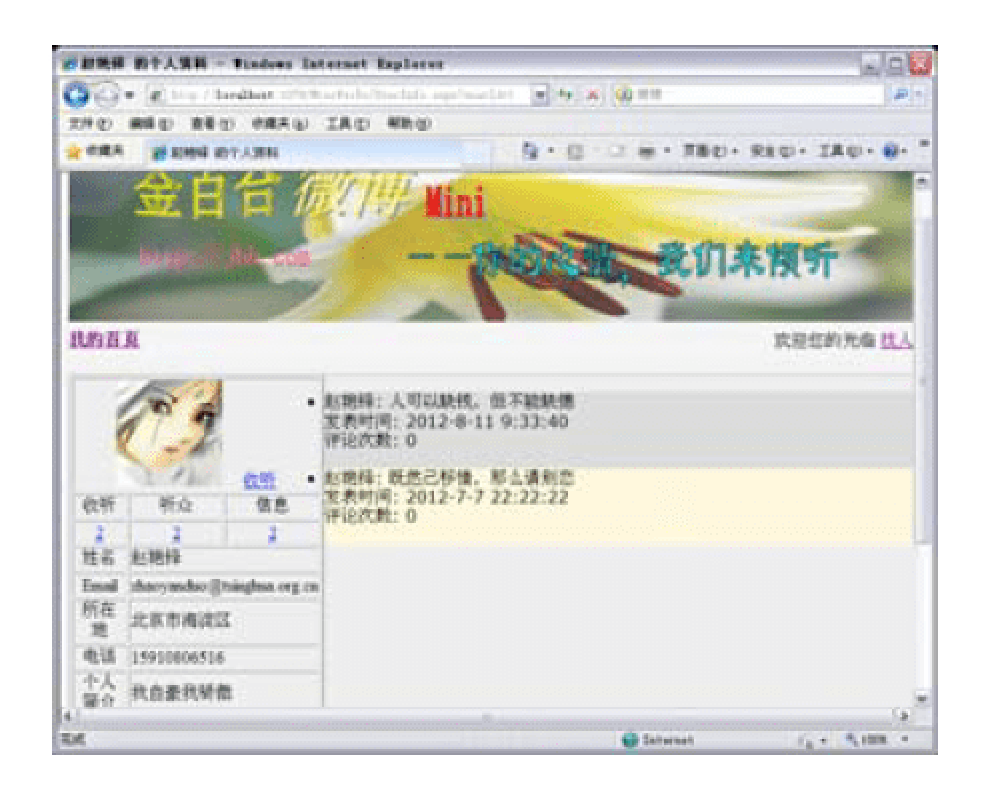


图 12-12　个人资料页面



图 12-13　收听用户后刷新了首页信息

单击某条消息后面的“转播”链接，将跳转到该消息的转播页面，在此可输入新的评论进行转播，如图 12-14 所示。

返回首页，单击“维护个人信息”链接，可打开个人信息维护页面，在此页中，可修改自己的信息，上传新的个性头像，如图 12-15 所示。



图 12-14　转播消息



图 12-15　个人信息维护

系统的其他功能这里就不一一列举了，读者可自行上机验证。

参 考 文 献

[1] 杨建军. ASP.NET 3.5 动态网站开发实用教程. 北京：清华大学出版社，2010.

[2] [美]Imar Spaanjaars 著. ASP.NET 4 入门经典(第 6 版). 刘伟琴，张格仙，译. 北京：清华大学出版社，2010.

[3] 梁灿，赵艳铎. Access 数据库应用基础教程. 北京：清华大学出版社，2005.

[4] 韩颖. ASP.NET3.5 动态网站开发实用教程. 北京：清华大学出版社，2011.

[5] 王辉，来羽，陈德祥. ASP.NET 3.5(C#)实用教程. 北京：清华大学出版社，2011.

[6] [美]David M.Kroenk，David J.Auer 著. 数据库原理(第 5 版). 赵艳铎，葛萌萌，译. 北京：清华大学出版社，2011.

[7] 杨春元. ASP.NET4.0 动态网站开发实用教程. 北京：清华大学出版社，2012.

[8] 蒋培，王笑梅. ASP.NET Web 程序设计. 北京：清华大学出版社，2007.

[9] 杨云，王毅. ASP.NET 2.0 程序开发详解. 北京：人民邮电出版社，2007.

[10] 程不功，龙跃进，卓琳. ASP.NET 2.0 动态网站开发教程. 北京：清华大学出版社，2007.

[11] 赵艳铎. 网页制作三剑客(MX 2004 版)精彩实例详解. 上海：上海科学普及出版社，2004.

[12] 肖建. ASP.NET 编程基础. 北京：清华大学出版社，2004.

[13] 冯方. ASP.NET 上机练习与提高. 北京：清华大学出版社，2007.

[14] 芦扬. Visual C# 2010 程序设计实用教程. 北京：清华大学出版社，2012.

[15] 王愉. DHTML 动态网页设计. 北京：人民邮电出版社，2007.

[16] http://www.asp.net

[17] http://jquery.com/